A Formal Temporal Semantics for Microsoft Project based on Allen's Interval Algebra

Denis Gagné

Trisotech Inc.
3100 Côte vertu, B380, Montreal, Canada.
dgagne@trisotech.com

André Trudel

Acadia University
Wolfville, Nova Scotia, Canada
Andre.Trudel@acadiau.ca

Abstract

Process modelling systems are complex and difficult to compare. A key attribute of any process modelling formalism or tool is time which involves how it handles and represents temporal dependencies and constraints. We are interested in doing a temporal based comparison between process modelling formalisms and tools by first converting them to a common representation. The temporal representation chosen is Allen's interval algebra. In this paper, we explain how to convert a project specified in Microsoft Project to a set of logical formulas. This conversion provides a formal temporal semantics for Microsoft Project.

Introduction

Business processes require the coordinated execution of individual activities to achieve a common business goal. A process model formally describes the structure of the workflow and how these atomic activities are coordinated and enacted by central or distributed workflow enactment services [WFMC 1995].

There exists a variety of process modeling formalisms and tools allowing one to capture process models at different levels of abstraction and addressing various business and-or technological concerns.

In recent years, many research initiatives have been dedicated to defining various workflow patterns. Each of the resulting pattern frameworks provides process constructs from a different perspective such as control [van der Aalst et. al. 2003], data [Russell et. al. 2004a], resources [Russell et. al. 2004b], and exceptions [Russell et. al. 2006]. These pattern frameworks are useful for:

- supporting business process modeling efforts,
- as a reference for learning basic and complex process concepts, and
- to assess and compare the expressiveness of various formalisms and tools.

We are primarily interested in the last use above but, from a temporal perspective. We do not use the traditional pattern approach of defining recurring temporal constructs within processes and then using these constructs to compare process models. We instead propose to compare process models in a more meaningful manner on the basis of formal semantics.

For each process modeling formalism or tool, we plan to convert its temporal constructs into first order temporal logic formulas derived from Allen's interval algebra. The resulting set of formulas will provide semantics for the process modeling formalism or tool. From there, the set of formulas derived from one formalism or tool will be compared against sets of formulas derived from other formalisms or tools.

Microsoft Project (MS Project) is not a process modeling tool *per se*, as it cannot capture some of the basic constructs of process models such as decision (or-split) and loops. Nevertheless, MS Project can still be used to model certain simple classes of processes. For example, we can capture in MS Project all activities related to the opening of a new store. We can then use this MS Project file as a franchise template by instantiating the series of coordinated activities captured in the MS Project file for all future store openings.

In this paper, we convert a project specified in MS Project to a set of logical formulas. We show how to convert each of MS Project's temporal constructs to a formula in Allen's interval algebra. One advantage of our semantics is that the formulas clarify some unexpected patterns allowed by MS Project. As a direct result of our deeper understanding of MS Project's semantics, we were able to correctly semantically parse MS Project files as process model inputs for a workflow management system which is currently under development. In our ongoing research, we are using the formulas to compare MS Project from a temporal perspective to various process modelling formalisms and tools.

The management of time in the context of business processes has received some attention in the past few years (e.g. [Combi et. al. 2002, 2003, 2004, 2006; Eder et. al. 2000, 2001; Lu et. al. 2006; Meyer et. al. 2003, 2004; Zhao et. al. 1999]). But, none of these previous papers deals with the problem of capturing the temporal constraints within a specific process modelling tool that is currently used and popular in the real world. Another difference between this paper and previous work in the area is the adoption of Allen's relations. Although Allen's interval algebra has been applied to many application areas, it has not received much attention in the business process community. To the best of our knowledge, only [Lu et. al. 2006] uses Allen's algebra as the basis for flexible business process execution via constraint satisfaction.

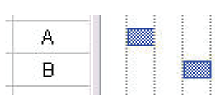
The remainder of this paper is organized as follows. After providing a brief overview of Allen's interval algebra, we introduce various temporal constructs in MS Project. For each construct, we provide a brief overview, followed by its semantics based on Allen's interval algebra. We conclude with an example.

Allen's Interval Algebra

We represent a temporal interval A by its endpoints written as (A^-, A^+). For example, the interval A = (1,10) has a left endpoint $A^- = 1$, and right endpoint $A^+ = 10$. All intervals are convex.

The underlying temporal structure is assumed to be discrete, linear, and totally ordered. An example of such a structure is the integers.

The most popular temporal reasoning approach in Artificial Intelligence is due to Allen [Allen 1983]. Allen's approach is based on intervals and the 13 possible binary relations between them. The first relation is "precedes" which is represented by the letter "p". Interval A precedes interval B if A ends before B starts (i.e., $A^+<B^-$), and is written as "A p B". If A precedes B, then it is also the case that B is preceded by A. This inverse relation for precedes, "proceeded by", is represented by "pi". The proceeds relation is shown at the top of figure 1. The diagram for precedes shows interval A to the left of interval B:



The diagram's representation is shown to its right and left. In this case, we have "A p B" and its equivalent inverse notation "B pi A".

The other relations are meets (m), overlaps (o), during (d), starts (s), finishes (f), and equals (e). As with proceeds, each of these relations has an inverse which is represented by appending an "i" to the relation symbol: mi, oi, di, si, and fi. The inverse of equals is equals. We refer to the 13 relations as the basic labels and they are all shown in figure 1.

Allen's interval relations are mutually exhaustive. For example, given two intervals, exactly one of the 13 relations will hold between them. It is impossible to have none or, two or more relations true between two temporal intervals.

Often, there is uncertainty as to exactly which relation holds between two intervals. For example, supper may be before or after going to the movies. We write this as:

(supper p movies) xor (supper pi movies)

Since exactly one of "p" or "pi" will be true, we use an exclusive-or (i.e., "xor"). A shorthand notation for the above formula is:

supper {p, pi} movies

In general, the relation between two intervals is allowed to be any subset of I = {p,pi,m,mi,o,oi,d,di,s,si,f,fi,e} including I itself.

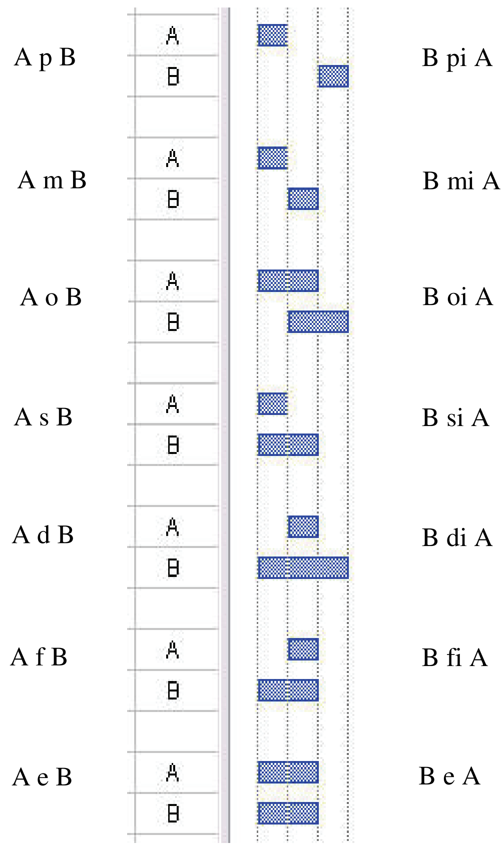


Figure 1: Allen's 13 relations

MS Project

MS Project [Microsoft Project 2007] is a product within the Microsoft Office Suite that allows the user to specify, plan and schedule various types of projects. A feature of MS Project is that it can calculate a realistic schedule for a project based simply on task durations and task dependencies. Other basic scheduling controls within MS Project include the Start and Finish dates of tasks, and the various available Calendars. MS Project uses four types of calendars: the base calendar, project calendar, resource calendar, and task calendar.

When a new project is created, the user must specify whether MS Project is to schedule the project from a specified start or finish date. As a default, MS Project will assume that the project is to be scheduled from the project start date, and that the start date is the current date at the time the new project is created.

Tasks within MS Project are entered in a task list. For example, tasks A, B, and C in figure 2. These tasks can be further organized and structured with the *Summary Tasks* feature. For example, the summary task at the top of figure 2 contains sub-tasks A, B, and C. Summary tasks can be used to summarize, show, or hide subtasks (e.g., compare the top expanded view to the bottom collapsed view of the same summary task in figure 2).

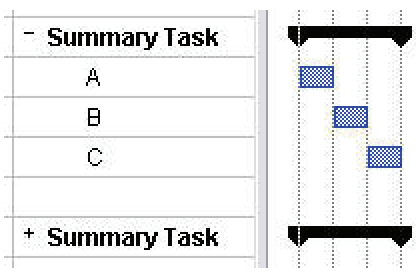


Figure 2: An expanded and collapsed summary task with subtasks.

Placing tasks in such a hierarchical structure does not create or imply any task dependencies among themselves. It is possible for a pair of tasks, whether they contain subtasks or not, to be related by any of Allen's thirteen basic interval relations. For example, even though tasks A and B in figure 2 are drawn as if there is a meets relationship between them, this is not necessarily the case. Any of the 13 relations can hold between A and B.

The sub-tasks within a summary task must occur during the summary task. For example, the summary task in figure 2 can only terminate after A, B, and C have all terminated.

Note that upon creation of a Task, MS Project by default assigns an "As Soon As Possible" start constraint to the newly created task (this constraint is discussed later).

Non trivial projects contain many tasks which depend on one another. A *Task Dependency* is a relationship between two tasks in which one task depends on the start or finish of another task in order to begin or end. The task that depends on the other task is the *Successor*, and the task it depends on is the *Predecessor*. Together, these two types of tasks help bind and give structure to a project.

Task dependencies allow MS Project to shift (recalculate) the schedule of all successor tasks whenever the start, finish or duration of any predecessor task is changed. The shift usually has a cascade effect throughout the project.

Task dependencies can be further refined with delays called *Lead and Lag Time. Lag Time* can be used to specify a delay between the finish of the predecessor and the start of the successor task. For example, we need a delay between the finish of the task "Paint wall" and the start of the next task "Hang pictures" to allow the paint to dry. Lag time can be specified either as a duration, such as 1 day, or as a percentage of the predecessor's duration, such as 25%. For example, if "Paint wall" has a 4 day duration, entering 1day or 25% would result in a 1 day delay to allow the paint to dry before starting the task "Hang pictures".

Lead Time causes the overlap of two tasks. In this case, the successor starts before the predecessor finishes. Lead time is useful when a successor task requires a head start. Lead time is entered as negative lag, such as:

-1 day or -25%.

For example, for the tasks "Construct walls" and "Plaster walls," we can use lead time to begin "Plaster walls" when "Construct walls" is half done.

In MS Project, both lag and lead time are specified using a single value. A positive lag specifies a Lag Time, a negative lag specifies a Lead Time, and a zero lag specifies that there is neither a Lag nor a Lead Time.

Temporal semantics

In this section, we provide a temporal semantics for MS Project based on Allen's relations. Specifically, we give an axiomatization for summary tasks, constraints on individual tasks, and constraints between tasks. First, we introduce a function and some constants.

Lag is always specified relative to two tasks. We represent the lag as a binary function. The function *lag(A,B)* returns the lag between tasks A and B. For example, a lag of 5 between tasks A and B is written as lag(A,B)=5. If lag(A,B) is negative, it represents a lead time.

We represent the project's start and finish date with the constants *project-start-date*, and *project-finish-date* respectively. Note that these constants represent a point and not an interval.

For summary tasks, assume a summary task A contains sub-tasks A1, A2,…, An, then each sub-task occurs during A:

$$\forall Ai \quad A^- \leq Ai^- \leq Ai^+ \leq A^+ \qquad \textbf{(ST)}$$

Note that one or more sub-tasks can start or end at the same instant as the summary task A.

MS Project allows the user to specify temporal constraints between tasks, and also apply them to an individual task. In the next sub-section, we describe the possible temporal constraints between two tasks. For each, we provide a definition followed by a first order logical representation using Allen's Interval Algebra. The second sub-section does the same with the temporal constraints applied to a single task.

Temporal relationships between tasks

Temporal relationships between two tasks in MS Project are represented with links between the tasks. The links can be used to specify four types of task dependencies:

1. Finish-to-start (FS),
2. Finish-to-finish (FF),
3. Start-to-start (SS), and
4. Start-to-finish (SF).

Each of the four types of dependencies is described below. For each, we also provide a formal semantics based on Allen's interval algebra.

Finish-to-start (FS). A Finish-to-start link or dependency between tasks A and B means that B cannot start until A finishes. For example, the task "Paint fence" cannot start until "Construct fence" finishes. This is the most common type of dependency and is pictured at the top of figure 3.

The Finish-to-start semantics is not obvious. When the lag variable is zero, we can have a meets relationship between the tasks as shown at the top of figure 3. But, MS Project also allows the user to move either task A to the left or move B to the right to create a precedes relationship between the tasks. Note that in this case the lag variable remains at zero. The Finish-to-start semantics in terms of Allen's interval algebra is:

$$lag(A,B)=0 \rightarrow A\{p,m\}B \quad \textbf{(FS)}$$

We can specify a Finish-to-start relationship with a Lag Time by setting the lag variable to a positive value. This has the effect of separating the tasks and the predecessor is always to the left of the successor. For example, if we change the lag variable from zero to 5, we have the situation shown in the middle of figure 3. The relationship between tasks A and B is precedes. Either task can be shifted to increase the distance between them. Note that increasing the distance does not change the value of the lag variable nor the precedes relation between the tasks. When lag is set to 5, MS Project will not allow tasks A and B to be closer than 5 units apart. The semantics of a Finish-to-start with a Lag Time is:

$$lag(A,B)>0 \rightarrow [[B^- - A^+ \geq lag(A,B)] \ \& \ A\{p\}B] \quad \textbf{(FS lag)}$$

A negative lag variable is used to specify a Finish-to-start relationship with a Lead Time. The bottom diagram in figure 3 shows a Finish-to-start relationship with lag=-3. In this case, A is preceded by B. If we hold A fixed, B cannot be shifted to the left. But, B can be shifted anywhere to the right without altering the value of lag. The result is that any relationship is possible between A and B:

$$lag(A,B)<0 \rightarrow [[B^- \geq (A^+ - |lag(A,B)|)] \ \& \ AIB] \quad \textbf{(FS lead)}$$

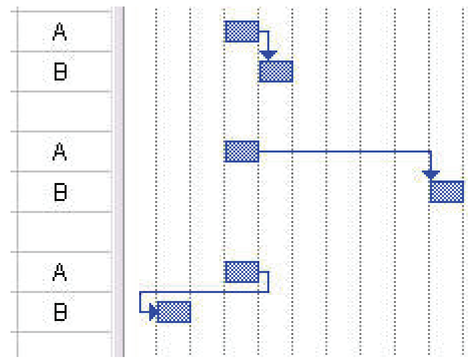


Figure 3: Finish-to-start dependencies, with Lag Time and Lead Time.

Start-to-start (SS). A Start-to-start link or dependency between tasks A and B means that B cannot start until A starts. For example, if A is "Pour foundation" and B is "Level concrete," "Level concrete" cannot begin until "Pour foundation" begins. The semantics for Start-to-start is:

$$lag(A,B)=0 \rightarrow A\{p,m,o,fi,di,s,e,si\}B \quad \textbf{(SS)}$$

$$lag(A,B)>0 \rightarrow [[B^- - A^- \geq lag(A,B)] \ \& \ A\{p,m,o,fi,di\}B] \quad \textbf{(SS lag)}$$

$$lag(A,B)<0 \rightarrow [[B^- \geq (A^- - |lag(A,B)|)] \ \& \ AIB] \quad \textbf{(SS lead)}$$

Finish-to-finish (FF). A Finish-to-finish link or dependency between tasks A and B means that B cannot finish until A finishes. For example, if A is "Add wiring" and B is "Inspect electrical," "Inspect electrical" cannot finish until "Add wiring" finishes. Note that with a Finish-to-finish dependency, B can finish later than A's finish time. The semantics is:

$$lag(A,B)=0 \rightarrow A\{p,m,o,fi,s,e,d,f\}B \quad \textbf{(FF)}$$

$$lag(A,B)>0 \rightarrow [[B^+ - A^+ \geq lag(A,B)] \ \& \ A\{p,m,o,s,d\}B] \quad \textbf{(FF lag)}$$

$$lag(A,B)<0 \rightarrow [[B^+ \geq (A^+ - |lag(A,B)|)] \ \& \ AIB] \quad \textbf{(FF lead)}$$

Start-to-finish (SF). A Start-to-finish link or dependency between tasks A and B means that B cannot finish until A starts. This dependency type can be used for just-in-time scheduling up to a milestone or the project finish date to minimize the risk of a task finishing late if its dependent tasks slip. The semantics is:

$$\text{lag}(A,B)=0 \rightarrow [A\ \text{I-}\{pi\}\ B] \qquad \textbf{(SF)}$$

$$\text{lag}(A,B)>0 \rightarrow [[B^+ - A^- \geq \text{lag}(A,B)]\ \&\ [A\ \text{I–}\{pi,mi\}\ B]] \qquad \textbf{(SF lag)}$$

$$\text{lag}(A,B)<0 \rightarrow [[B^+ \geq (A^- - |\ \text{lag}(A,B)\ |)]\ \&\ A I B] \qquad \textbf{(SF lead)}$$

Constraints on an Individual Task

Constraints can also be specified to control the start or finish date of a task. These constraints can be flexible (not tied to a specific date) or inflexible (tied to a specific date).

Flexible Constraints. A *Flexible Constraint* does not specify a specific date for a task. The flexible constraints are:

- As Soon As Possible (ASAP),
- As Late As Possible (ALAP),
- Finish No Earlier Than (FNET),
- Finish No Later Than (FNLT),
- Start No Earlier Than (SNET), and
- Start No Later Than (SNLT).

Flexible constraints work in conjunction with task dependencies to make a task occur as soon or as late as the task dependency will allow. For example, a successor task with an As Soon As Possible (ASAP) constraint and a finish-to-start dependency will be scheduled as soon as the predecessor task finishes.

Constraints with moderate scheduling flexibility restrict a task from starting or finishing before or after a specified date. For example, a successor task with a Start No Later Than (SNLT) constraint of June 15 and a finish-to-start dependency can begin any time its predecessor is finished up until June 15, but can't be scheduled after June 15.

The flexible constraints are described in detail below.

As Soon As Possible (ASAP)

If a task is assigned an As Soon As Possible constraint, MS Project schedules the task as early as it consistently can. No additional date restrictions are put on the task. This is the default constraint for newly created tasks in projects scheduled from the project start date. Note that the task will not be scheduled by MS Project before project-start-date. The semantics is:

$$\text{minimize}(A^-)\ \ \&\ \ A^- \geq \text{project-start-date} \qquad \textbf{(ASAP)}$$

As Late As Possible (ALAP)

This constraint is analogous to the ASAP constraint, but relative to the end of the project. If a task is assigned an As Late As Possible constraint, MS Project schedules the task as late as it consistently can, given other scheduling parameters. No additional date restrictions are put on the task. This is the default constraint for newly created tasks in projects scheduled from the project finish date. If you choose to schedule your project from the project finish date, MS Project will determine how late you can start your project and still finish by the specified project finish date. The semantics is:

$$\text{maximize}(A^+)\ \ \&\ \ A^+ \leq \text{project-finish-date} \qquad \textbf{(ALAP)}$$

Finish No Later Than (FNLT)

This constraint indicates the latest possible date that this task is to be completed. It can be scheduled to finish on or before the specified date. A predecessor task cannot push a successor task with an FNLT constraint past the constraint date. The semantics for "task A FNLT a specific date *d*" is:

$$A^+ \leq d \qquad \textbf{(FNLT)}$$

Note that every occurrence of "*d*" in the formula above and the remainder of this section is a point and not a temporal interval.

Start No Later Than (SNLT)

This constraint indicates the latest possible date *d* that this task can begin. The task can be scheduled to start on or before the specified date. A predecessor task cannot push a successor task with an SNLT constraint past the constraint date. For projects scheduled from the finish date, this constraint is applied when a start date is entered for a task. The semantics is:

$$A^- \leq d \qquad \textbf{(SNLT)}$$

Finish No Earlier Than (FNET)

This constraint indicates the earliest possible date *d* that this task can be completed. The task cannot be scheduled to finish any time before the specified date. For projects scheduled from the start date, this constraint is applied when a finish date is entered for a task. The semantics is:

$$A^+ \geq d \qquad \textbf{(FNET)}$$

Start No Earlier Than (SNET)

This constraint indicates the earliest possible date *d* that a task can begin. The task cannot be scheduled to start any time before the specified date. For projects scheduled from the start date, this constraint is applied when a start date is entered for a task. The semantics is:

$$A^{-} \geq d \qquad \textbf{(SNET)}$$

Inflexible Constraints. An Inflexible Constraint ties a task to a date. In MS Project, the inflexible constraints are:

- Must Finish On (MFO) and
- Must Start On (MSO).

Inflexible Constraints override any task dependencies and restrict a task to a date. For example, a task with a Must Start On (MSO) constraint for September 30 and a finish-to-start dependency to another task will always be scheduled for September 30 whether its predecessor finishes early or late. The inflexible constraints are described below.

Must Start On (MSO)
This constraint indicates the exact date on which a task must be scheduled to begin. Other scheduling parameters such as task dependencies, lead or lag time and delay cannot affect scheduling the task unless this requirement is met. The semantics for "task A has an MSO of date *d*" is:

$$A^{-} = d \qquad \textbf{(MSO)}$$

Must Finish On (MFO)
This constraint indicates the exact date *d* on which a task must be scheduled to be completed. Other scheduling parameters such as task dependencies, lead or lag time, resource leveling, and delay cannot affect scheduling the task unless this requirement is met. The semantics is:

$$A^{+} = d \qquad \textbf{(MFO)}$$

Example

A simple example of an MS Project is shown in figure 4. We use this example to show how the formulas in the previous sections can be iteratively applied to provide a temporal semantics.

The example in Figure 4 captures a simple software development process where A stands for the "Inception" phase, B is the "Elaboration" phase, C is the "Construction" phase and D is the "Transition" phase. Further details of the "Elaboration" phase B are provided: B1 stands for "Complete Analysis", B2 is "Prepare Use Cases", B3 is "Use Cases Review Meeting", B4 is "Risks Analysis", and B5 and B6 are "Design Model" and "Design Review Meeting" respectively.

We derive the set of formulas (**EX-1-20**) for this example by iteratively applying the temporal formulas introduced in this paper. The example was drawn with a lag of zero between tasks and sub-tasks:

lag(A,B)=0 (**EX-1**)

lag(B,C)=0 (**EX-2**)

lag(C,D)=0 (**EX-3**)

lag(B1,B2)=0 (**EX-4**)

lag(B2,B3)=0 (**EX-5**)

lag(B3,B5)=0 (**EX-6**)

lag(B4,B5)=0 (**EX-7**)

lag(B5,B6)=0 (**EX-8**)

Note that the link between B4 and B5 in figure 4 may be misleading. It appears as if the lag is 3. It was indeed specified as zero.

There is a finish-to-start dependency between summary tasks A, B, C and D. From **(EX-1-3)** and **(FS)** we derive:

A{p,m}B (**EX-9**)

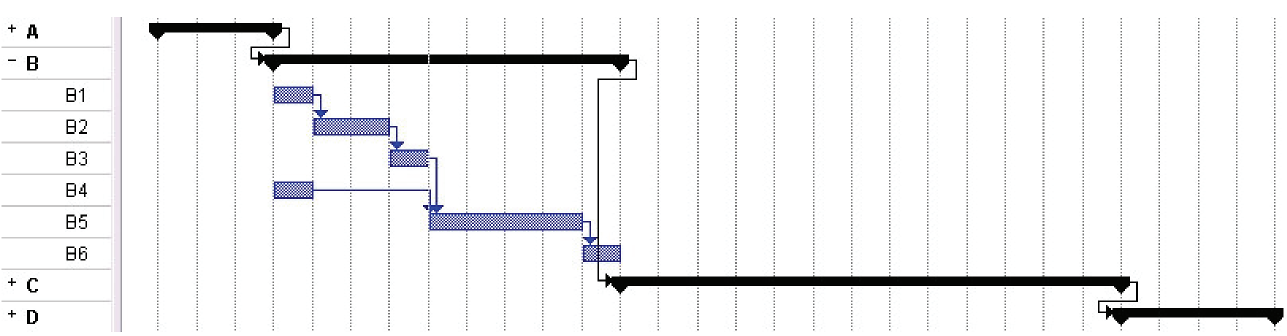


Figure 4: Finish-to-start dependencies, with Lag Time and Lead Time.

$$B\{p,m\}C \quad \textbf{(EX-10)}$$

$$C\{p,m\}D \quad \textbf{(EX-11)}$$

We derive similar formulas for the sub-tasks B1-B6:

$$B1\{p,m\}B2 \quad \textbf{(EX-12)}$$

$$B2\{p,m\}B3 \quad \textbf{(EX-13)}$$

$$B3\{p,m\}B5 \quad \textbf{(EX-14)}$$

$$B4\{p,m\}B5 \quad \textbf{(EX-15)}$$

$$B5\{p,m\}B6 \quad \textbf{(EX-16)}$$

Using axiom (ST), we capture the temporal relationships between the summary task B and its sub-tasks:

$$\forall Bi \quad B^- \leq Bi^- \leq Bi^+ \leq B^+ \quad \textbf{(EX-17)}$$

Optionally, we can explicitly represent the fact that there is no known temporal relationship between B4 and B1, B2, B3:

$$B1 \; I \; B4 \quad \textbf{(EX-18)}$$

$$B2 \; I \; B4 \quad \textbf{(EX-19)}$$

$$B2 \; I \; B4 \quad \textbf{(EX-20)}$$

Conclusion & Future Work

There are many diverse process modeling formalisms and tools. Since they are often based on different paradigms, they are difficult to compare. Our long term goal is to compare these process modeling formalisms and tools based on their temporal capabilities.

Rather than simply identifying temporal modeling constructs or patterns, we have opted to compare process models in a more meaningful manner on the basis of formal semantics. The advantage of this approach is that it leads to a deeper semantical notion of process model equivalence rather than simple comparison of expressiveness based on constructs.

Another advantage of our approach is that once a formal semantics has been specified for a particular project, the resulting formulas can be checked for consistency (e.g., an off the shelf constraint satisfaction package can be used). We may also be able to derive interesting conclusions from the formulas.

For our temporal formalism, we chose Allen's which is the most popular in Artificial Intelligence. As for a first process modelling framework, we chose MS Project because it is the most widely used and known project planning tool in industry. Although MS project is not a process modeling tool *per se* and its temporal constraints seem relatively simple, the study of their semantics leads to interesting results which are not immediately evident from naïve usage or simply reading the documentation provided.

As a direct result of our deeper understanding of MS Project's semantics, we were able to correctly semantically parse MS Project files as process model inputs for a workflow management system which is currently under development.

Acknowledgments. The second author is supported by an NSERC Discovery Grant.

References

van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., and Barros, A.P. 2003. Workflow Patterns. *Distributed and Parallel Databases,* 14(3): 5-51.

Allen, J.F. 1983. Maintaining Knowledge about Temporal Intervals, *Communications of the ACM 26:* 832-843.

Combi, C., and Pozzi, G. 2002. Towards Temporal Information Workflow Systems. In *ER 2002* 13-25. Berlin: Volume 2784, *Lecture Notes in Computer Science*, Springer-Verlag.

Combi, C., and Pozzi, G. 2003. Temporal Conceptual Modelling of Workflows. In *ER 2003*, 59-76. Berlin: Volume 2813, *Lecture Notes in Computer Science*, Springer-Verlag.

Combi, C., and Pozzi, G. 2004. Architectures for a Temporal Workflow Management System. In *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC'04)*, Nicosias, Cyprus.

Combi, C., and Pozzi, G. 2006. Task Scheduling for Temporal Workflow Management System. In *Proceedings of the Thirteenth International Symposium on Temporal Representation and Reasoning (TIME'06)*: IEEE.

Eder, J., Gruber, W., and Panagos, E., 2000. Temporal Modeling of Workflows with Conditional Execution Paths, In *Database and Expert Systems Applications*, 243-253: Springer.

Eder, J., and Paganos, E., 2001. Managing Time in Workflow Systems. In *Workflow Handbook 2001*, Layna Fischer (Ed.), Future Strategies Inc., USA.

Lu, R., Sadiq, S., Padmanabhan, V., and Governatori, G. 2006. Using a Temporal Constraint Network for Business Process Execution. In *Proceedings of the 17th Australian Database Conference(ADC2006)*. Hobart, Australia:

Volume 49, Conferences in Research and Practice in Information Technology (CRPIT).

Meyer, A., McGough, S., Furmento, N., Lee, W., Newhouse, S., and Darlington, J., 2003. ICENI Dataflow and Workflow: Composition and Scheduling in Space and Time. In *Proc of UK e-Science All Hands Meeting*, EPSRC.

Meyer, A., McGough, S., Furmento, N., Lee, W., Gulamali, M., Newhouse, S., and Darlington, J., 2004. Workflow Expression: Comparison of Spatial and Temporal Approaches. In *Workflow in Grid Systems Workshop (GGF-10)*, Berlin.

Microsoft Project, Online Documentation. Retrieved April 2, 2007. http://office.microsoft.com/enca/assistance/CH790018101033.aspx

Russell, N., ter Hofstede, A.H.M., Edmond, D., and van der Aalst, W.M.P. 2004a. Workflow Data Patterns, QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane.

Russell, N., ter Hofstede, A.H.M., Edmond, D., and van der Aalst, W.M.P. 2004b. Workflow Resource Patterns, BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven.

Russell, N., van der Aalst, W.M.P., and ter Hofstede, A.H.M. 2006. Workflow Exception Patterns. In *Proceedings of the 18th International Conference on Advanced Information Systems Engineering (CAiSE'06)*, 288-302. Berlin: Volume 4001, *Lecture Notes in Computer Science*, Springer-Verlag.

WFMC 1995. The Workflow Reference Model, WFMC-TC-1003, 1.1.

Zhao, J.L., and Stohr, E.A., 1999. Temporal Workflow Management in a Claim Handling Systems. In *Proceedings of the international joint conference on Work activities coordination and collaboration (WACC '99).* 187-195, New York: ACM Press.