Mapping Timepoint-Based Constraints into Interval Relationships

Stephen T. Polyak

Artificial Intelligence Applications Institute (AIAI)
Institute for Representation and Reasoning (IRR)
Division of Informatics, The University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, United Kingdom
Steve_Polyak@ed.ac.uk

Abstract

In our work on a Common Process Framework (CPF) we have sought to define a life-cycle of activities surrounding the management of process knowledge. One of the most important aspects of process knowledge involves time. Different views of time may be expressed with either timepoints or time intervals. The primary representation of time we use in CPF involves the former while we are also interested in mapping this information to the latter. Unique sets of timepoint-based constraints between two process activities may be legal or illegal (i.e. they are consistent or inconsistent) and they may also either partially or completely specify a relationship from Allen's standard set of 13. In this paper we explore this space of mappings between timepoint-based constraints and interval relationships and report on the number of legal, complete and partial constraint configurations.

Introduction

In our work on a Common Process Framework (CPF) we have sought to define a life-cycle of activities surrounding the management of process knowledge. Our approach is centred around knowledge-rich processes which are based on past and present work in Artificial Intelligence (AI) planning and plan representations. In addition to this, we have also gained insight through our involvement with standards work related to the exchange of process knowledge.

At the heart of this framework is a process ontology, the Common Process Ontology (CPO) (Polyak 1998). The ontology defines terms and concepts which can be used to describe the space of process behavior from a design-based perspective. CPO is based on a constraint model of activity (Tate 1996a) in which processes may be designed by adding various types of constraints including temporal, resource, state-based, etc.

In the design of an ontology there are typically several choices in representing and defining various concepts. An ontology describing plans or processes is certainly no exception. One of the most important ontological design decisions in developing such an ontology is its formalisation of time. Various levels of commitment can be made to the structure of time. As a general "rule of thumb" though we note that high levels of commitment restrict the applicability of the ontology while lower levels of commitment may offer general terms and definitions which can be specialized as needed.

A fundamental issue though involving an ontological commitment to time revolves around a basic stance on what we *mean* by "time". At least two of the possible senses embody either **timepoints** or **time intervals**. The following excerpts from (Hayes 1996) outlines these two approaches. Regarding intervals

> "[They are] ... pieces of time; physical entities whose sole dimension is time-dimension. These are variously called time-periods or time-intervals, or simply intervals. Examples include during the 1994 winter Olympics, the sixteenth century and 10:50 to 11:00 a.m. on 30 May 1993. These are particular pieces of time located in (or perhaps, parts of) the time-plenum. Intervals are in many ways the most central concept for temporal reasoning since they are the temporal extents of things. Events typically are thought of as occupying them, propositions are true during them and they are the lifetimes of objects."

A timepoint-based approach on the other hand is distinguished as well

> "[Another] notion is that of a timepoint. Exactly what counts as a point, and the relationships between points and intervals, seem to be particularly controversial and sensitive questions, and many of the formalisations in use in computer science have taken one or another stance on the answers to these questions."

There are times when we may prefer to work with timepoints and there are other times when time inter-

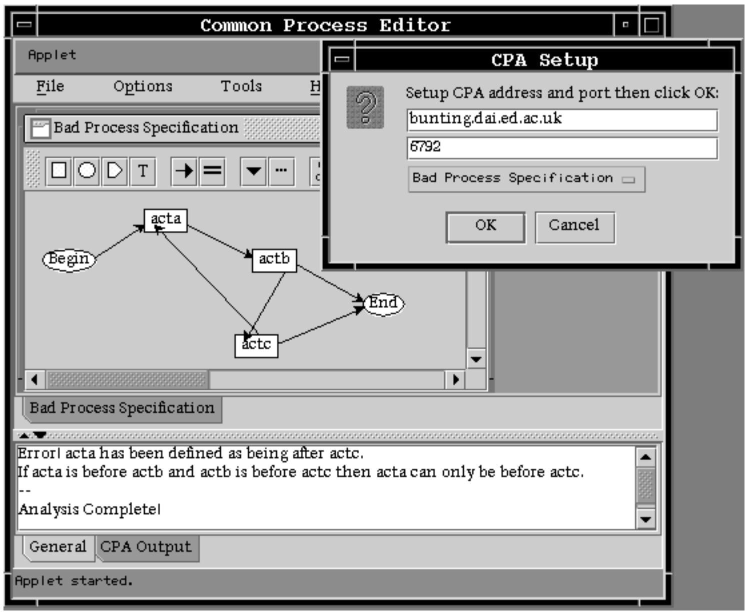

Figure 1: An example of a CPE error.

vals are more appropriate. For example, in specifying a process we may wish to have a highly flexible approach in which we can attach begin and end timepoints to activities and then order those timepoints using "before" or "equals" relations. As a result of evaluating such a network though we may wish to explain errors in a particular specification designed with such constraints from an interval perspective rather than a point-based perspective (e.g. we would expect something like "activity A cannot be after activity B given that etc." rather than "the end point of activity B cannot be ordered before the begin point of activity A given that etc.").

In fact, this particular scenario is enacted within the CPF. The java-based domain and process editors (Common Domain Editor, CDE and Common Process Editor, CPE) permit interaction with node ends of activities which are associated with timepoints. When the user chooses to evaluate the process network, it is sent to a Prolog-based Common Process Assistant (CPA) via TCP/IP. The CPA attempts to map this timepoint-based knowledge into interval-based knowledge so that it can use interval-based reasoning to detect illegal configurations. The CPA then reports back errors, rationale, and suggestions from an interval-based perspective.

It is the intent of this paper to explore the CPA mapping of the actions and their respective timepoints onto time intervals and to illustrate how this knowledge could be used to prevent illegal or unnecessary constraints in the editors.

CPO ⇒ CPA

(Tate 1996b) developed an informal plan ontology based on the <I-N-OVA> constraint model of activity cited above. CPO is very closely aligned with these sentence-level descriptions. In this approach, a timepoint-based schema is advocated.

A timepoint in CPO, $\mathcal{T}p$, characterises a specific, instantaneous point that lies along a line which is an infinite sequence of time points. Conceptually, pairs of timepoints for nodes and processes delimit a time interval. So given that, how do we map sets of temporal constraints on timepoints into interval relationships? First, we must enumerate the possible set of interval relationships that we will be interested in. For this, we turn to Allen's definition of the 13 relations between intervals (Allen 1984a; 1984b) which is typically taken to be the standard set. This set of relationships includes the following six **{before, meets, overlaps, starts, during, finishes}** along with their inverses and **equals**.

We can utilise an axiomatisation based on Hayes' catalogue of temporal theories (Hayes 1996) in order to map timepoints and ordering constraints into these 13 relationships. In fact, during our application of these definitions, we spotted and corrected a couple of errors in the source mapping axiomatisation. The axioms are listed in the appendix.

As we have indicated, this axiomatisation is used in the Common Process Assistant (CPA) to map the timepoints and ordering constraints which are passed from the process and domain editing tools, upon a users request, into an interval theory for consistency checking. Allen's table of legal relationships between intervals (Allen 1984a) is then used to detect errors and to provide rationale for why a process specification is incorrect (i.e. CPA explains which legal interval relationships could exist). For example, Figure 1 shows a simple Common Process Editor specification which was passed to the CPA. In CPE the left half of the node is used to graphically indicate the begin timepoint while constraint attachments to the right half indicate relationships assigned to its end timepoint.

The CPA output message area displays the information passed back from CPA which includes the set of errors along with an explanation and suggestion of possible corrections for each error. In this case there is only one error. It should be noted that the nature of this reported error depends on the order in which the constraints are processed. For example, the error could also have been reported to be between B and C or A and B. When searching for errors, CPA maintains a set of known errors which are a triple, <i,j,k>, in which the relationships between intervals i and j and j and k conflicts with a relationship between k and i. When it realizes that there is an error already reported involving the intervals i, j, and k it refrains from repeating the error using different bindings.

Analysis: Timepoints and Intervals

In this section, we explore the relationship between timepoints and time intervals in a bit more detail. In particular, we are interested in the implications of our approach whereby temporal constraints on timepoints are incrementally added to an activity specification and how that affects a mapping to time interval relationships between 2 activities. Two important perspectives we address are: the validity of a set of temporal constraints; and the cases in which those constraints completely or partially specifies some Allen relationship between the intervals implied by the activities. For this analysis we will outline some basic terms and notations.

Each activity, $\mathcal{A}$, has 2 timepoints which we will abbreviate as: $\mathcal{T}p^{\mathcal{A}}_{begin}$ and $\mathcal{T}p^{\mathcal{A}}_{end}$. There is one relation that always exists between an activity, A_1, timepoint pair: $before\_tp(Tp^{A_1}_{begin}, Tp^{A_1}_{end})$. No other relation can be made between these two points. For any two different activities in a process activity specification, $\mathcal{A}s$, there is a set of unique pairs of timepoints, which we will refer to as a $\mathcal{T}set^{\mathcal{A},\mathcal{A}}$, which is defined

$$
\begin{aligned}
Tset^{A_1,A_2} \quad = \quad & \{(tp_1, tp_2) | tp_1 \in \{Tp^{A_1}_{begin}, Tp^{A_1}_{end}\} \wedge \\
& tp_2 \in \{Tp^{A_2}_{begin}, Tp^{A_2}_{end}\}\}
\end{aligned}
$$

Each pair in a $\mathcal{T}set$ may be related in one of two possible ways or not at all. If a relationship is assigned to a pair then either one timepoint is temporally before the other or they are equal. We will express this in an infix notation as

$$
\begin{aligned}
tp_1 \prec tp_2 \quad &\equiv \quad before\_tp(tp_1, tp_2) \\
tp_1 = tp_2 \quad &\equiv \quad equal\_tp(tp_1, tp_2)
\end{aligned}
$$

Given this information, we can see that there are 256 unique configurations of relationships between this set of pairs. This calculation is arrived at in the following way

$$
\begin{array}{rr}
1 & 0-assignment \\
+12 & 1-assignment \\
+(12*9)/2 & 2-assignment \\
+(12*9*6)/6 & 3-assignment \\
\underline{+(12*9*6*3)/24} & 4-assignment \\
256 & unique-combinations
\end{array}
$$

For this calculation, the space of combinations is divided into four sets: the set containting 0 user-specified constraints between the timepoints (0-assignment); the set containing 1 user-specified constraint (1-assignment); 2 (2-assignment); etc. up to four constraints. Note that once a single constraint between two timepoints has been selected, it reduces the possible set by 3, not 1. For example, if a user specifies that $(Tp^{A_1}_{begin} \prec Tp^{A_2}_{begin})$ then it also eliminates $(Tp^{A_2}_{begin} \prec Tp^{A_1}_{begin})$ and $(Tp^{A_1}_{begin} = Tp^{A_2}_{begin})$ as well.

Of course these are not all legal combinations of constraints. An example of an illegal 2-assignment combination would be $\{(Tp^{A_2}_{end} \prec Tp^{A_1}_{begin}), (Tp^{A_1}_{begin} \prec Tp^{A_2}_{begin})\}$. Clearly, if A_2 is before A_1, which is what the first constraint says, then it cannot be the case that $Tp^{A_1}_{begin}$ is before $Tp^{A_2}_{begin}$.

0-assignment

We could start by looking at these constraints level by level. In the first level (0-assignment) there is only one possible configuration which corresponds to no user-specified assignments being made between any of the $\mathcal{T}set$ elements. Note that we say user-specified, because there are two implied constraints always present: $\{(Tp^{A_1}_{begin} \prec Tp^{A_1}_{end}), (Tp^{A_2}_{begin} \prec Tp^{A_2}_{end})\}$ which comes from the CPO ontology.

1-assignment

Stepping down a level, we can see that there are twelve unique, single assignment configurations between any $(tp_1, tp_2) \in Tset^{A_1,A_2}$. This is very straight forward as we can see that there are a total of 12 unique constraints which could be assigned. These include

$$\begin{array}{rcl}
\{(Tp^{A_1}_{begin} \prec Tp^{A_2}_{begin})\} & \vee & \{(Tp^{A_1}_{begin} \prec Tp^{A_2}_{end})\} \vee \\
\{(Tp^{A_1}_{end} \prec Tp^{A_2}_{begin})\} & \vee & \{(Tp^{A_1}_{end} \prec Tp^{A_2}_{end})\} \vee \\
\{(Tp^{A_2}_{begin} \prec Tp^{A_1}_{begin})\} & \vee & \{(Tp^{A_2}_{begin} \prec Tp^{A_1}_{end})\} \vee \\
\{(Tp^{A_2}_{end} \prec Tp^{A_1}_{end})\} & \vee & \{(Tp^{A_2}_{end} \prec Tp^{A_1}_{begin})\} \vee \\
\{(Tp^{A_1}_{begin} = Tp^{A_2}_{begin})\} & \vee & \{(Tp^{A_1}_{begin} = Tp^{A_2}_{end})\} \vee \\
\{(Tp^{A_2}_{begin} = Tp^{A_1}_{end})\} & \vee & \{(Tp^{A_2}_{end} = Tp^{A_1}_{end})\}
\end{array}$$

In appendix B we provide tables and figures which outline our results of the analysis at each level (> 0). For example, Table 1 lists the 12 1-assignment configurations. The following substitutions have been made to make the table more legible $B_1 = Tp^{A_1}_{begin}$, $E_1 = Tp^{A_1}_{end}$, $B_2 = Tp^{A_2}_{begin}$, $E_2 = Tp^{A_2}_{end}$. The entries in the right-most column index into the figure above it (i.e. Figure 2). This figure graphically displays these unique, legal configurations and separates them into partial and complete sets. We note that all 12 possible configurations are legal and 4 configurations completely describe an Allen relationship between the intervals and the remaining 8 do not, which we call "partial" configurations.

Before we move on to explore the **2-assignment**, **3-assignment**, and **4-assignment** sets we will provide a generic description of the procedure we are following to analyze this space of possible configurations.

Analysis Procedure

The following algorithm defines the procedural steps we are following to inspect the space of possible timepoint-based constraint configurations. In this algorithm we are interested in deriving a count of legal specifications (ls), partial specifications (ps), complete specifications (cs), and the list of legal, unique constraint sets (ucs).

1 PROCEDURE analyse-configurations()
2 BEGIN
3   ls $\leftarrow$ 1;
4   ps $\leftarrow$ 1;
5   cs $\leftarrow$ 0;
6   ccs $\leftarrow \{(Tp^{A_1}_{begin} \prec Tp^{A_1}_{end}), (Tp^{A_2}_{begin} \prec Tp^{A_2}_{end})\}$;
7   pcs $\leftarrow$ possible-constraint-set();
8   ucs $\leftarrow$
9    $\{\{(Tp^{A_1}_{begin} \prec Tp^{A_1}_{end}),$
10    $(Tp^{A_2}_{begin} \prec Tp^{A_2}_{end})\}\}$;
11   analyse-subroutine(ccs);
12   PROCEDURE analyse-subroutine (ccs$'$)
13   BEGIN
14   FOR c $\leftarrow$ EACH ELEMENT OF pcs
15   DO BEGIN
16   IF (legal($\{$c $\cup$ ccs$'\}$)
17   AND $\{$c $\cup$ ccs$'\} \notin$ ucs)
18   THEN BEGIN
19   ls $\leftarrow$ ls + 1;
20   ucs $\leftarrow \{$c $\cup$ ccs$'\} \cup$ ucs;
21   IF (complete($\{$c $\cup$ ccs$'\}$)) cs $\leftarrow$ cs + 1;
22   ELSE ps $\leftarrow$ ps + 1;
23   IF (count($\{$c $\cup$ ccs$'\}$) $<$ 4)
24   analyse-subroutine($\{$c $\cup$ ccs$'\}$);
25   END;
26   END;
27   END;
28 END;

In lines 3-10 we initialise a number of variables to setup the call to the recursive analyse-subroutine procedure. Note that this procedure is defined in a similiar way to a standard ML local function. By this we mean to indicate that the procedure is not restricted to using parameters and local variables, it may freely access variables that are defined within its scope.

The legal specification counter (ls) and partial specification counter (ps) are both initialized to 1 to represent the 0-configuration we discussed above. It is obvious then that there are no complete specifications (cs=0) at this stage (0-assignment). We initialize the current constraint set (ccs) to include those implied by the CPO ontology. The possible constraint set (pcs) is an unchanging set which contains the 12 we introduced above:

29 FUNCTION possible-constraint-set (): SET
30 BEGIN
31   return(
32    $\{(Tp^{A_1}_{begin} \prec Tp^{A_2}_{begin}), (Tp^{A_1}_{begin} \prec Tp^{A_2}_{end}),$
33    $(Tp^{A_1}_{end} \prec Tp^{A_2}_{begin}), (Tp^{A_1}_{end} \prec Tp^{A_2}_{end}),$
34    $(Tp^{A_2}_{begin} \prec Tp^{A_1}_{begin}), (Tp^{A_2}_{begin} \prec Tp^{A_1}_{end}),$
35    $(Tp^{A_2}_{end} \prec Tp^{A_1}_{end}), (Tp^{A_2}_{end} \prec Tp^{A_1}_{begin}),$
36    $(Tp^{A_1}_{begin} = Tp^{A_2}_{begin}), (Tp^{A_1}_{begin} = Tp^{A_2}_{end}),$
37    $(Tp^{A_2}_{begin} = Tp^{A_1}_{end}), (Tp^{A_2}_{end} = Tp^{A_1}_{end})\}$);

38 END;

Finally, the list of unique constraint sets (ucs) is initialised to include one element, the single configuration in the 0-assignment level. Note that while ccs may be assigned a set of constraints, ucs is actually a set of sets of constraints.

The local analyse-subroutine procedure accepts the current constraint set as a parameter (line 12). For each possible constraint, the algorithm determines if the union of the selected constraint and the current constraint set is legal (i.e. the set of constraints do not lead to a contradiction and there are no repeats). In addition to this, the ucs is consulted to see if this proposed configuration has already been identified. If the proposed set is legal and unique the counter is incremented and the configuration is added into the ucs.

It is then evaluated to see if it completely specifies an Allen relationship. This can be determined by using the axiomatization in the appendix. The appropriate counter is then incremented (lines 21 or 22). Finally, if the number of constraints in this new configuration does not equal the maximum (4) then we must evaluate the possibility of adding additional constraints. The analyse-subroutine is called recursively with the new constraint set. When the analyse-configurations procedure is complete, we should have all of the desired information acquired at line 28.

1-assignment (revisited)

We observed that there were 12 unique, legal configurations for the 1-assignment set. Four of those 12 "completely specify" an Allen interval relationship (along with the inverses)., viz.

$$
\begin{aligned}
Tp_{end}^{A_1} \prec Tp_{begin}^{A_2} &\equiv before(A_1, A_2) \\
Tp_{end}^{A_2} \prec Tp_{begin}^{A_1} &\equiv after(A_1, A_2) \\
Tp_{end}^{A_1} \prec Tp_{begin}^{A_2} &\equiv meets(A_1, A_2) \\
Tp_{end}^{A_2} \prec Tp_{begin}^{A_1} &\equiv met-by(A_1, A_2)
\end{aligned}
$$

The remaining 8 1-assignment configurations can be considered to specify "partial" Allen interval relationships, that is to say that they cannot yet map into any of the identified 13 relationships given the axiomatization. As we mentioned before, this is summarized in Table 1 and Figure 2.

2-assignment

Table 2 and Figure 3 summarizes the validity of the relationship between C_1 and $Fig2$ for the 2-assignment level. By C_1 mean some constraint which was selected by line 14. The x-axis of Table 2 is populated by the set of legal specifications which were derived from the previous level (i.e. 1-assignment in this case). Since this is a reflexive relation only half of the table needs to be generated. Legal pairs of assignments are referenced by the number which indexes into the graphical presentation in Figure 3. In addition to this, we have placed a '×' to indicate points at which the test at line 16 failed. Failure of the test at line 16 is also indicated by '•' which means that the proposed constraint was a repeat of one already in the current constraint set. As you can see from Table 2 there are configurations which map into the same, unique specification (i.e. they share the same index). This is detected at line 17.

We can see from Figure 3 that there are actually 18 unique, legal 2-assignment configurations. Ten of them are "complete" Allen interval relationships and the remaining 8 are still "partial". At this level, only the *overlaps* and its inverse $overlapped-by$ relationship from Allen's set of 13 cannot be expressed. These remaining relationships require at least a 3-assignment configuration.

[3,4]-assignment

At 3-assignment, we have 14 configurations which are "complete" Allen interval relations and 5 are "partial" for a total of 19 unique, legal 3-assignment specifications (see Table 3 and Figure 4). Note that while the y-axis of the tables remains the same, the x-axis depends on the legal specifications uncovered in the level above it. In this case the Figure 3 specifications.

Finally, at 4-assignment we have 7 "complete" and 0 "partial" configurations (Table 4 and Figure 5) corresponding to all of the 13 interval relationships (given the inverses).

Results and Discussion

As we can see, only 57 of the 256 unique combinations are legal ($\approx 22\%$) and only 35 of these 57 completely specifies an Allen interval relationship ($\approx 7\%$ of the total). This analysis provides knowledge we can use to construct efficient process editors which prevent users from specifying illegal configurations between two activities (while relying on the CPA interval reasoning to check transitive relationships). Consider the case where some user-defined constraint, C_1, already exists between a pair $(tp_1, tp_2) \in Tset^{A_1,A_2}$. Given this constraint between (tp_1, tp_2) (or given some set of constraints) we can encode a table lookup which tells us what other constraint, C_2, can be assigned to a pair, $(tp_3, tp_4) \in Tset^{A_1,A_2}$, whereby C_2 is still consistent with the assertion from C_1.

In addition to this, it is obvious that some of the configurations contain superfluous constraints which can be safely eliminated. For example, the $Tp^{A_j}_{end} \prec Tp^{A_i}_{end}$ constraint in the specification labelled "17" in Figure 3 is unnecessary as we can infer this relation given that $(Tp^{A_i}_{begin} = Tp^{A_j}_{end}) \wedge (\mathcal{T}p^{\mathcal{A}}_{begin} \prec \mathcal{T}p^{\mathcal{A}}_{end})$.

Acknowledgements

The author is sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-96-1-0348 – an AASERT award monitored by Dr. Abe Waksman and associated with the O-Plan project F30602-97-1-0022. The U.S. Government is authorised to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of DARPA,AFOSR or the U.S. Government.

References

Allen, J. 1984a. Maintaining knowledge about temporal intervals. *Commun. ACM* 26(11):832–843.

Allen, J. 1984b. Towards a general theory of action and time. In Allen, J.; Hendler, J.; and Tate, A., eds., *Readings in Planning*. Palo Alto, CA: Morgan Kaufmann. 464–479.

Hayes, P. 1996. A catalog of temporal theories. Technical Report Technical Report UIUC-BI-AI-96-01, The Beckman Institute, University of Illinois.

Polyak, S. 1998. A common process ontology for process-centred organisations. Department of Artificial Intelligence DP ??, University of Edinburgh, Edinburgh, Scotland.

Tate, A. 1996a. Representing plans as a set of constraints – the < I-N-OVA > model. In Drabble, B., ed., *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, 221–228. Edinburgh, Scotland: Morgan Kaufmann.

Tate, A. 1996b. Towards a plan ontology. *AI*IA Notiziqe (Publication of the Associazione Italiana per l'Intelligenza Artificiale), Special Issue on Aspects of Planning Research* 9(1):19–26.

A. Interval Definitions for CPA

This axiomatisation is based on Hayes' definitions (Hayes 1996) of the isomorphic relationship between data structures in a timepoint-based theory and an interval-based theory (Allen 1984a; 1984b).

$$\begin{aligned}(\forall a).(before(begin-timepoint(a),\\ end-timepoint(a)))\\ \supset timeinterval(a)\end{aligned}$$

$$\begin{aligned}(\forall tp_1 tp_2).(tp_1 = begin-timepoint(between(tp_1,tp_2)) \wedge\\ tp_2 = end-timepoint(between(tp_1,tp_2)))\\ \Leftrightarrow before(tp_1,tp_2)\end{aligned}$$

$$\begin{aligned}(\forall a_1 a_2).(timeinterval(a_1) \wedge timeinterval(a_2) \wedge\\ before(end-timepoint(a_1), begin-timepoint(a_2)))\\ \Leftrightarrow precedes(a_1,a_2)\end{aligned}$$

$$\begin{aligned}(\forall a_1 a_2).(timeinterval(a_1) \wedge timeinterval(a_2) \wedge\\ before(begin-timepoint(a_1), begin-timepoint(a_2)) \wedge\\ before(begin-timepoint(a_2), end-timepoint(a_1)) \wedge\\ before(end-timepoint(a_1), end-timepoint(a_w)))\\ \Leftrightarrow overlaps(a_1,a_2)\end{aligned}$$

$$\begin{aligned}(\forall a_1 a_2).(timeinterval(a_1) \wedge timeinterval(a_2) \wedge\\ begin-timepoint(a_1) = begin-timepoint(a_2) \wedge\\ before(end-timepoint(a_1), end-timepoint(a_2)))\\ \Leftrightarrow starts(a_1,a_2)\end{aligned}$$

$$\begin{aligned}(\forall a_1 a_2).(timeinterval(a_1) \wedge timeinterval(a_2) \wedge\\ before(begin-timepoint(a_2), begin-timepoint(a_1)) \wedge\\ before(begin-timepoint(a_1), end-timepoint(a_1))) \wedge\\ before(end-timepoint(a_1), end-timepoint(a_2)))\\ \Leftrightarrow during(a_1,a_2)\end{aligned}$$

$$\begin{aligned}(\forall a_1 a_2).(timeinterval(a_1) \wedge timeinterval(a_2) \wedge\\ before(begin-timepoint(a_2), begin-timepoint(a_1)) \wedge\\ end-timepoint(a_1) = end-timepoint(a_2))\\ \Leftrightarrow finishes(a_1,a_2)\end{aligned}$$

B. Analysis Results, Tables and Figures

The following tables and figures summarize the results of the analysis procedure which was outlined on page 4. The table entries index into a graphical presentation in the figure above it. Each table/figure pairing represents a level in the space of possible timepoint-based configurations.

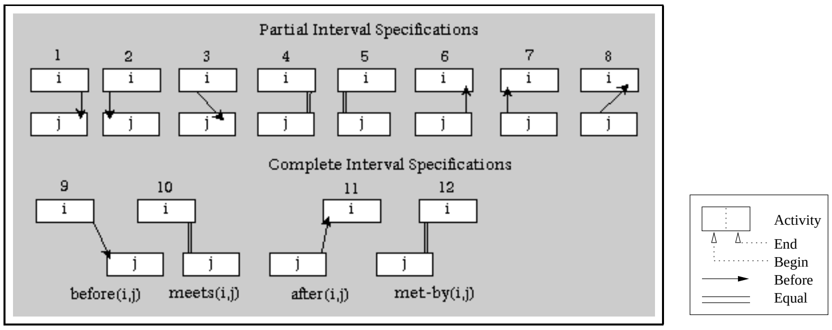

Figure 2: Legal and unique 1-assignment specifications

C_1	
$E_1 \prec B_2$	9
$B_1 \prec E_2$	3
$B_1 \prec B_2$	2
$E_1 \prec E_2$	1
$B_1 = B_2$	5
$E_1 = E_2$	4
$B_1 = E_2$	12
$E_1 = B_2$	10
$E_2 \prec B_1$	11
$B_2 \prec E_1$	8
$B_2 \prec B_1$	7
$E_2 \prec E_1$	6

Table 1: Possible set of 1-assignment constraint specifications

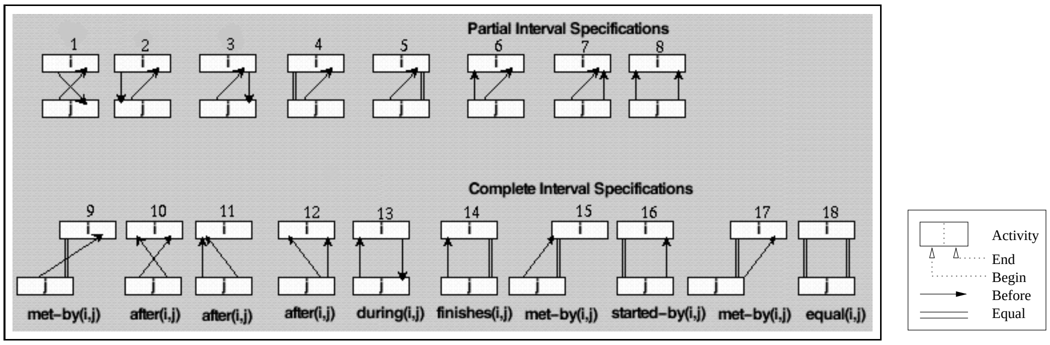

Figure 3: Legal and unique 2-assignment specifications

$C_1/Fig2$	11	8	7	6	5	4	12	10	9	3	2	1
$E_1 \prec B_2$	×	×	×	×	×	×	×	×	•	10	11	12
$B_1 \prec E_2$		1	2	3	4	5	9	×	10	•	6	7
$B_1 \prec B_2$			×	13	×	14	15	×	11	6	•	8
$E_1 \prec E_2$				×	16	×	17	×	12	7	8	•
$B_1 = B_2$					•	18	×	×	×	4	×	16
$E_1 = E_2$						•	×	×	×	5	14	×
$B_1 = E_2$							×	•	×	×	×	×
$E_1 = B_2$								×	×	9	15	17
$E_2 \prec B_1$									×	×	×	×
$B_2 \prec E_1$										1	2	3
$B_2 \prec B_1$											×	13
$E_2 \prec E_1$												×

Table 2: Validity of 2-assignment constraints, ×=illegal,$<num>$=legal,•=repeat

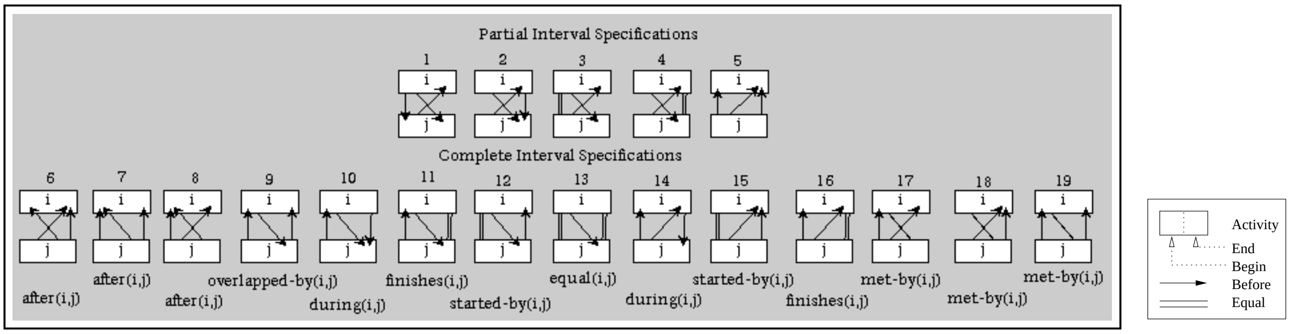

Figure 4: Legal and unique 3-assignment specifications

$C_1/Fig3$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$E_1 \prec B_2$	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
$B_1 \prec E_2$	•	1	2	3	4	1	2	9	×	×	×	×	10	11	×	12	×	13
$B_1 \prec B_2$	1	•	9	×	11	×	10	×	×	×	×	×	×	×	×	×	×	×
$E_1 \prec E_2$	•	9	×	12	×	14	×	×	×	×	×	×	•	×	×	×	×	×
$B_1 = B_2$	3	×	12	•	13	×	15	×	×	×	×	×	×	×	×	•	×	•
$E_1 = E_2$	4	11	×	13	•	16	×	×	×	×	×	×	×	•	×	×	×	•
$B_1 = E_2$	×	×	×	×	×	17	18	19	•	×	×	×	×	×	•	×	•	×
$E_1 = B_2$	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
$E_2 \prec B_1$	×	×	×	×	×	8	6	7	×	•	•	•	×	×	×	×	×	×
$B_2 \prec E_1$	•	•	•	•	•	•	•	9	•	•	8	6	14	16	17	15	18	13
$B_2 \prec B_1$	1	×	14	×	16	•	5	•	17	8	•	7	•	•	•	×	19	×
$E_2 \prec E_1$	2	10	×	15	×	5	•	•	18	6	7	•	×	×	19	•	•	×

Table 3: Validity of 3-assignment constraints, $\times$=illegal,$<num>$=legal,$\bullet$=repeat

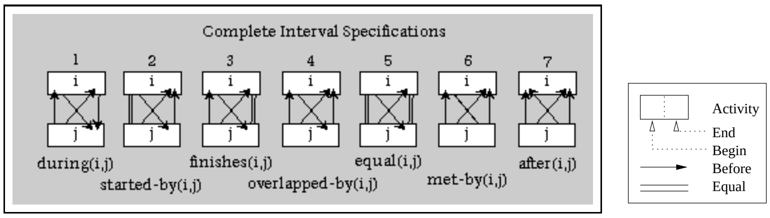

Figure 5: Legal and unique 4-assignment specifications

$C_1/Fig4$	1	2	3	4	9	10	11	12	13	14	15	16	17	18	19	8	6	5	7
$E_1 \prec B_2$	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
$B_1 \prec E_2$	•	•	•	•	•	•	•	•	•	1	2	3	×	×	×	×	×	4	×
$B_1 \prec B_2$	•	4	×	3	•	•	•	×	×	×	×	×	×	×	×	×	×	×	×
$E_1 \prec E_2$	4	•	2	×	×	•	×	•	×	•	×	×	×	×	×	×	×	×	×
$B_1 = B_2$	×	2	•	5	×	×	×	•	•	×	•	×	×	×	×	×	×	×	×
$E_1 = E_2$	3	×	5	•	×	×	•	×	•	×	×	•	×	×	×	×	×	×	×
$B_1 = E_2$	×	×	×	×	×	×	×	×	×	×	×	×	•	•	•	×	×	6	×
$E_1 = B_2$	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
$E_2 \prec B_1$	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	•	•	7	•
$B_2 \prec E_1$	•	•	•	•	•	1	3	2	•	•	•	•	•	•	6	•	•	•	7
$B_2 \prec B_1$	×	1	×	3	•	•	•	×	×	•	×	•	•	6	•	•	7	•	•
$E_2 \prec E_1$	1	•	2	×	•	•	×	•	×	×	•	×	6	•	•	7	•	•	•

Table 4: Validity of 4-assignment constraints, $\times$=illegal,$<num>$=legal,$\bullet$=repeat