– TIP: AN INTERMEDIATE LANGUAGE FOR THE DESCRIPTION OF TEMPORAL SITUATIONS –

ESPRIT 2434

Contribution to:

Activity Task 2.5

-- System Concept for Advanced CIM/AI Controller --

Title:

TIP: AN INTERMEDIATE LANGUAGE FOR THE DESCRIPTION OF TEMPORAL SITUATIONS

Author/Company/Copyright:

Jörg - Ingo Jakob

Achievements:

TIP (acronym from Temporal Inference Propagation), a new language for the description of temporal situations, is presented. It incorporates the work of Allen, Rit and Becker on qualitative and quantitative representation of temporal situations and expresses this work in a unifying language approach, enhanced by arithmetic and procedural constructs. TIP is presented in a formal manner, providing a syntax expressed in BNF. TIP can be used to describe a temporal situation declaratively offline, or to communicate with the underlying TIP inference engine interactively online. Several application examples of TIP are given, amongst them 'real life' applications from the domains of planning and scheduling.

Summary:

A language for the description of temporal situations (called TIP) is presented. It incorporates the work of Allen, Rit and Becker on qualitative and quantitative representation of temporal situations and expresses this work in a unifying language approach, enhanced by arithmetic and procedural constructs. TIP is intermediate in the sense that it is a kind of assembler language for the declarative description of temporal situations. By careful choice of TIP's syntax, mnemonics and structure, TIP allows its application as an interpreted interface language between user and the underlying TIP inference engine as well. Several application examples are provided, amongst them 'real life' applications from the planning and scheduling domains. An outlook on possible additional language features in future versions of TIP is given.

Index

TIP:
AN INTERMEDIATE LANGUAGE FOR THE DESCRIPTION OF TEMPORAL SITUATIONS

0. Introduction

TIP (Temporal Inference Propagation / Propagator) is not only a program, but provides a language for the description of temporal "situations" as well[1]. This paper intends to define the syntax and semantics of TIP's language, as provided by the forthcoming version 2.0 of TIP.

It is assumed that the reader / user of TIP has a background in the field of temporal representation techniques, and especially has some knowledge of the particular work done by Allen et al. [Allen 1983] [Rit 1986] [Becker 1988].

Throughout the rest of this paper, TIP is synonymous with "TIP's language"; in contrast to the program TIP itself, which is then referred to as the 'TIP program', 'TIP system' or 'TIP shell'.

1. TIP Situation File Structure

A TIP situation can be provided as a file, or can be fed to the TIP program by its input stream (which can be regarded as a file as well).

Any temporal situation described by TIP is composed of lines. Each line starts with a unique keyword out of a set of keywords (cf. chapter 2). Such a leading keyword specializes the syntax and semantics of the line. Empty lines or lines containing only blanks or tabs are allowed and are skipped.

Comments always occupy a complete line. In order to distinguish a line containig comments from an ordinary keyword line, a comment has to start with a semicolon in the first column of a line.

There is no explicit syntactical order in which keywords must appear throughout a situation. However, there is an implicit semantical order in which keywords may appear. For example, an edge referred to in an IF statement must exist. Thus it must have previously been generated by RESTRICT or CONNECT. Below the syntax is given formally in a BNF-like meta-language (cf. [Wirth 1974]); the semantical side is given verbally only.

2. TIP's Language Elements (Keywords and Symbols)

Each individual situation is composed of lines, where each line starts with one of the following leading keywords:

Net
Restrict

[1]The term "situation" for the description of coherent temporal facts was originated by Allen [Allen 1983].

Sopo
Connect
Relax
Scope
If
Constrain
Holds
Situation
Propagate
Simplify
LMinimize
Minimize
Label
Clear
Exit

(A keyword is defined as an unique symbol denoting a syntactical and/or semantical unit or a part thereof.)

Other keywords do exist. These keywords appear throughout a line, but not in a leading position:

Then
To
Duration.

Then there are symbols (these are not considered to be keywords) which appear throughout a line, but not in a leading position. These are the positive integers of a certain range, and the symbols for the thirteen primitive temporal relations as introduced by Allen [Allen 1983]. Cf. chapter 3.2.

3. Basic Concepts and Definitions

3.1 The Initial Empty Net

After TIP was invoked from the underlying operating system, it is in its empty or initial state. In this state, TIP consists of MaxNode unconnected nodes. A node is unconnected, when it has no temporal relation to any other node. Furthermore, the sopo attached to each initial node is unspecified. An unspecified sopo is assumed to extend from Zero to PlusInfinity. (For a definition of the constants MaxNode, Zero and PlusInfinity, cf. chapter 3.2)

3.2 Constants and Data Types

Definition of Constants:

Constants, as defined in this formal language description, are used throughout this description. Constants are not recognized by the TIP parser. Thus they are to be distinguished from another kind of constants, called symbols, which are recognized by the TIP parser.

MaxNode: the largest node recognized by TIP.
MaxEdge: the largest edge recognized by TIP.
The following equation holds always: MaxEdge = MaxNode * (MaxNode-1) / 2 .

Zero: absolute zero on the time scale.
PlusInfinity: virtual infinity on the time scale.

*: Asterisk is a special constant (and symbol) denoting an unspecified (unavailable) time point or time duration. It may be used in place of any point of time.

NoInfo: a special constant denoting the compound relation composed of all thirteen primitive relations.

In TIP 2.0:

- MaxNode and MaxEdge are machine-dependent. Current settings (1) for a PC/AT-compatible computer under MS/DOS is MaxNode = 63, (2) for a VAX workstation under VAX/VMS is MaxNode = 500.

- Zero = 0. Interpretation: the origin on the time scale.

- PlusInfinity = $(2^{31} - 1) / 2$. Interpretation: time is discretized; there are exactly PlusInfinity + 1 distinguishable time points on the time scale (including Zero).

Data Types (Defined by Sets of Constants):

EdgeBaseType = [1, MaxEdge].

NodeBaseType = [1, MaxNode].

RelationBaseType = { After, Before, Contains, During, Overlaps, Overlapped-By, Starts, Started-By, Finishes, Finished-By, Meets, Met-By, Equals } .

StringType : any string of character which does not contain whitespace characters and whose maximum string length is 72.

TimeType = [Zero, PlusInfinity] .

ExtendedTimeType = *TimeType* U { * } .

4. Leading Keywords

4.1 Net

Syntax: Net <FromNode> <ToNode> .

Where <FromNode> and <ToNode> are members of *NodeBaseType*; and <FromNode> <= <ToNode>.

Semantics: Introduces a new temporal net to the TIP shell. Claims all the nodes from <FromNode> to <ToNode> to be constituting nodes of this net. Clears all these nodes and all edges connecting to these or from these nodes; even if these nodes have previously been part of another temporal net (which thereby is destroyed). Quantitative meaning: the sopos of these nodes are unspecified (thus extending from zero to infinity on the time scale). Qualitative meaning: these nodes are currently not connected to any other nodes by means of edges.

4.2 Restrict

Syntax: Restrict <FromNode> <ToNode> <PrimitiveRelation>
{ <PrimitiveRelation>} .

Where <FromNode> and <ToNode> are members of *NodeBaseTye*; and where each <PrimitiveRelation> is a member of *RelationBaseType* and may appear once at most.

Semantics: (1) If the edge spanning between <FromNode> and <ToNode> had an "old" value: the value is restricted to the intersection of the old value and the new value consisting of all primitive relations appearing in this line. (2) If the edge spanning between <FromNode> and <ToNode> had no "old" value (i. e. this edge was undefined and did not previously exist): the edge is generated and assigned to span between <FromNode> and <ToNode>; its value is the relation consisting of all primitive relations appearing in this line.

Notice that in either case, no propagation of new edge values takes place (cf. keyword Propagate).

4.3 Sopo

Syntax: Sopo <Node> <EarliestStart> <LatestStart> <EarliestFinish> <LatestFinish>
<MinimalDuration> <MaximalDuration> .

Where <Node> is a member of *NodeBaseType*, and <EarliestStart>, <LatestStart>, <EarliestFinish>, <LatestFinish>, <MinimalDuration> and <MaximalDuration> are members of *ExtendedTimeType*, and (1) <EarliestStart> < <LatestStart> and <EarliestFinish> < <LatestFinish>, (2) <MinimalDuration> < <MaximalDuration> [, (3) the < property may be replaced by the <= property by setting the appropriate switch in TIP's configuration file].

Semantics: The sopo of node <Node> is assigned the 6-tupel (<EarliestStart> <LatestStart> <EarliestFinish> <LatestFinish> <MinimalDuration> <MaximalDuration>) .

Notes: No further restrictions apply (i. e. the sopo will be accepted even if it is not 'adjusted' in the sense that its durations do not properly match start and finish, in which case the sopo is adjusted implicitly by TIP).

4.4 Connect

Syntax: Connect <FromNode> To <ToNode> |
Connect <Node> { <Node> }'.

Where <FromNode>, <ToNode> and <Node> are members of *NodeBaseType*; and <FromNode> < <ToNode>.

Semantics: Each node enumerated implicitely (by using the variant using keyword 'To') or explicitely (by using the variant explicitely enumerating nodes) shall be 'connected' to each other node in the current scope (cf. keywords Net, Scope). 'Connecting' means that edges spanning from this node to any other node are alloted, and that the value of these edges is set to NoInfo.

Note: Keyword Connect is in a sense doing the opposite of keyword Relax.

4.5 Relax

Syntax: Relax <FromNode> To <ToNode> |
Relax <Node> { <Node> } .

Where <FromNode>, <ToNode> and <Node> are members of *NodeBaseType*; and <FromNode> < <ToNode>.

Semantics: Each node enumerated implicitely (by using the variant using keyword 'To') or explicitely (by using the variant explicitely enumerating nodes) shall be 'relaxed' from each other node in the current scope (cf. keywords Net, Scope). 'Relaxing' means that edges spanning from this node to any other node shall be disposed of., i. e. that these edge shall be unconnected from now on.

Note: Keyword Relax is in a sense doing the opposite of keyword Connect.

4.6 Scope

Syntax: Scope <FromNode> <ToNode> .

Where <FromNode> and <ToNode> are members of *NodeBaseType*; and <FromNode> <= <ToNode>.

Semantics: The current scope is set to the range extending from <FromNode> to <ToNode>. Effects the evaluation of the keywords Propagate and Find. Propagate and Find will only propagate inside a subnet of the total temporal net. This subnet is delimited by the scope previously set using the Scope keyword. Cf. Propagate and Find.

4.7 If

Syntax: If <IfEdge> <IfPrimitiveRelation>

Then <ThenEdge> <ThenPrimitiveRelation>
{ <ThenPrimitiveRelation> } |
If <IfEdge> <IfPrimitiveRelation>
Then <ThenNode> Duration <Minimal Duration> <Maximal Duration>

Where <IfEdge> and <ThenEdge> are members of *EdgeBaseType*, and <IfPrimitiveRelation> and <ThenPrimitiveRelation> are members of *RelationBaseType*, and <Minimal Duration> and <Maximal Duration> are members of *TimeType*.

Semantics: The If keyword implements a kind of 'demon' or 'active value' mechanism.
(1) First form: if the edge <IfEdge> assumes the value <IfPrimitiveRelation>, then the edge <ThenEdge> will be restricted to the compound temporal relation consisting of all <ThenPrimitiveRelations>.
(2) Second form: if the edge <IfEdge> assumes the value <IfPrimitiveRelation>, then the duration of node <ThenNode> will be restricted to the time interval specified by <Minimal Duration> <Maximal Duration>.

Note: These are procedural enhancements to the purely descriptive temporal logic as described by Allen et al. (cf. chapter 0. Introduction). Cf. keyword Constrain.

4.8 Constrain

Syntax: Constrain <Node1> <Node2> <Node3> Duration + .

Where <Node1>, <Node2> and <Node3> are members of *NodeBaseType* .

Semantics: Establishes an arithmetic constraint between the durations of <Node1>, <Node2> and <Node3>. The constraint is composite; to explain it we introduce the following two arithmetic functions id and ad: id(S) yields the minimal duration of a sopo S, ad(S) yields the maximal duration of a sopo S. Then, the following arithmetic constraints hold:

id(<Node1>) + id(<Node2>) = id(<Node3>), *and*
ad(<Node1>) + ad(<Node2>) = ad(<Node3>) .

Note: This is an arithmetic enhancement to the purely temporal constraint propagation mechanism as described by Allen et al. (cf. chapter 0. Introduction). Cf. keyword If.

4.9 Holds

Syntax: Holds <Node> <SimpleRelation> <Count>.

Where <Node> is one of *NodeBaseType*, <SimpleRelation> is one of *RelationBaseType*, and <Count> is one of *EdgeBaseType*.

Semantics: Adds a local constraint to <Node>. This constraint requires that exactly <Count> primitive relations emanating from <Node> are a <SimpleRelation>. For conceptual simplicity, it is assumed that the edges spanning from <Node> to all other nodes which are connected to <Node> are unidirectional in the sense that they always start at <Node>. [This is contrary to the TIP-internal representation of an edge's direction, which assumes that <FromNode> < <ToNode>].

Note: This keyword can be useful in describing a temporal situation using an integer programming style.

4.10 Situation

Syntax: Situation <SituationFileName> .

Semantics: Conveys the file named <SituationFileName> to the TIP parser for interpretation.

Note: During evaluation of the keyword Situation, another keyword Situation may be encountered by the TIP reader. Then, situations are evaluated recursively by the TIP reader. Thus, nesting of situation files is allowed.

4.11 Propagate

Syntax: Propagate .

Semantics: Finds all 'solutions' to the current temporal net as delimited by the current scope. A 'solution' is any instantiation of the net which is qualitatively and quantitatively consistent, and where each edge inside the current scope contains exactly one primitive relation as its current value.

Note: Finding all solutions of a the current net may take fractions of a second, or it may be intractable at all due to run time limitations. (If in doubt, limit the number of solutions-to-be-found to a number close to 1 in TIP's configuration file.)

4.12 Simplify

Syntax: Simplify .

Semantics: Propagates all constraints that are members of the current scope (cf. keyword Scope). As a result of this propagation, either an identical or new net results, or the net may be found to be inconsistent. In case of a new net, the compound relation on each edge is a subset of the compound relation on that edge before propagation started. Cf. keyword Scope.

Note: This propagation is purely qualitative (it does *not* make use of the quantitative information contained in sopos in order to prune the edge relation values). This propagation can be considered to be a pre-processing of the net before the keyword Propagate is applied. Thereby, the search space is reduced.

4.13 LMinimize

Syntax: LMinimize <Node>;

where <Node> is one of *NodeBaseType.*

Semantics: Minimizes the duration of node <Node> locally. To that end, all edges which extend from or to <Node> and whose relation are still compound and not yet unique are considered. Of those edges, the edge mimizing the duration of node <Node> is selected and made unique itself.

Note: LMinimze is a procedural enhancement to the purely temporal constraint propagation mechanism as described by Allen et al. (cf. chapter 0. Introduction). LMinimze assumes that if-then rules do exist which can possibly fire to minimze node <Node>. Cf. keyword If.

4.14 Minimize

Syntax: Minimize <Node>;

where <Node> is one of *NodeBaseType.*

Semantics: Minimizes the duration of this particular node. In effect, if more than one solution of the current temporal net is found, the solution is retained which minimizes node <Node>.

Note: In TIP 2.0, at most one node may be a candidate for minimizing a net. If another, subsequent keyword Minimize is encountered, a previous one is overruled.

4.15 Label

Syntax: Label <Node> <String>.

Where <Node> is one of *NodeBaseType* and <String> is one of *StringType.*

Semantics: Attaches a string <String> to node <Node>. In case a non-empty string was attached to a node this way, the string will always appear besides the node in all file output TIP will generate (example: solution file.)

4.16 Clear

Syntax: Clear.

Semantics: Clears (forgets) the complete temporal net, regardless of the current scope. After evaluation of Clear, the TIP shell is in the same state as when freshly invocated from the underlying operating system. Cf. chapter 3.1.

4.17 Exit

Syntax: Exit.

Semantics: Exits (leaves) TIP and returns control to the underlying operating system.

5. Symbols

As defined in chapter 3.2, symbols are constants which are recognized by the TIP interpreter.

Currently, the following special symbols are recognized:

- the thirteen primitive temporal relations as defined by Allen [Allen 1983]: After, Before, Contains, During, Overlaps, Overlapped-By, Starts, Started-By, Finishes, Finished-By, Meets, Met-By, Equals.

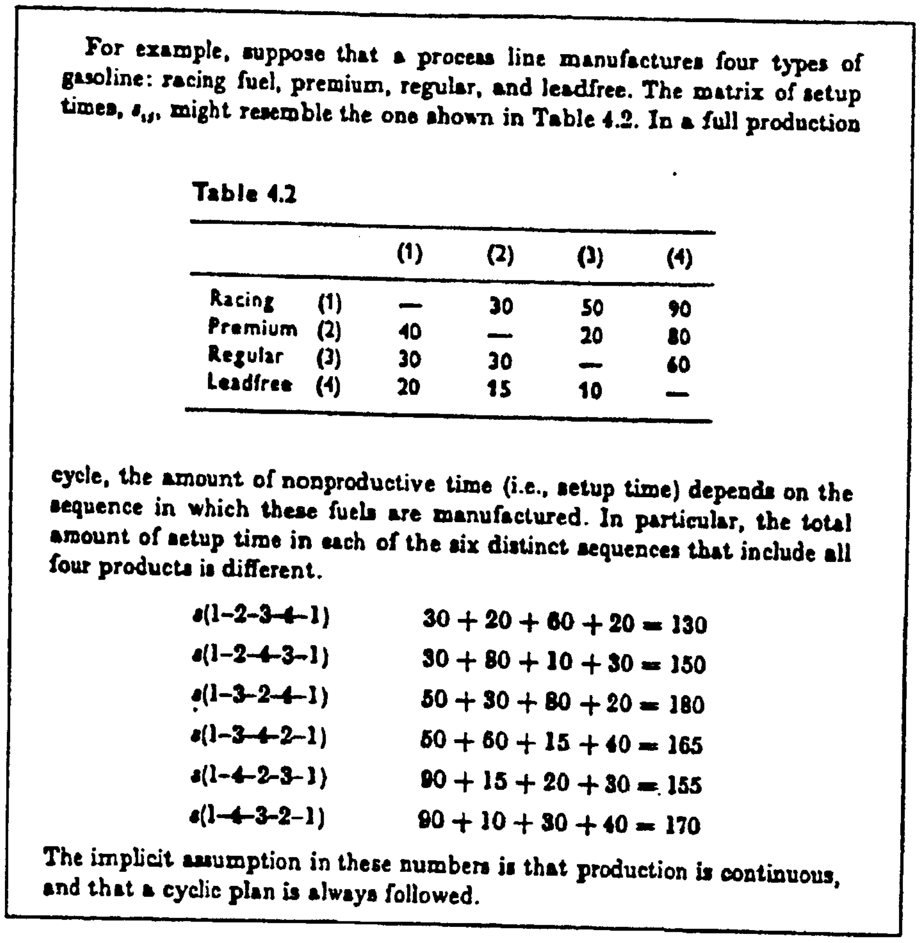
- the unspecified (unknown) point of time: *.

In addition to special symbols, all integers are recognized in the places proper defined by TIP's syntax, as described in this language reference.

6. TIP Application Examples

6.1 Example 1: Order Scheduling

The following example (figure 1) is taken from [Baker 1974], p. 95:



For example, suppose that a process line manufactures four types of gasoline: racing fuel, premium, regular, and leadfree. The matrix of setup times, s_{ij}, might resemble the one shown in Table 4.2. In a full production

Table 4.2

		(1)	(2)	(3)	(4)
Racing	(1)	—	30	50	90
Premium	(2)	40	—	20	80
Regular	(3)	30	30	—	60
Leadfree	(4)	20	15	10	—

cycle, the amount of nonproductive time (i.e., setup time) depends on the sequence in which these fuels are manufactured. In particular, the total amount of setup time in each of the six distinct sequences that include all four products is different.

$$s(1\text{–}2\text{–}3\text{–}4\text{–}1) \qquad 30 + 20 + 60 + 20 = 130$$
$$s(1\text{–}2\text{–}4\text{–}3\text{–}1) \qquad 30 + 80 + 10 + 30 = 150$$
$$s(1\text{–}3\text{–}2\text{–}4\text{–}1) \qquad 50 + 30 + 80 + 20 = 180$$
$$s(1\text{–}3\text{–}4\text{–}2\text{–}1) \qquad 50 + 60 + 15 + 40 = 165$$
$$s(1\text{–}4\text{–}2\text{–}3\text{–}1) \qquad 90 + 15 + 20 + 30 = 155$$
$$s(1\text{–}4\text{–}3\text{–}2\text{–}1) \qquad 90 + 10 + 30 + 40 = 170$$

The implicit assumption in these numbers is that production is continuous, and that a cyclic plan is always followed.

Figure 1: Baker's example with sequence-dependent setup times [Baker 1974]

A possible formal description using the TIP language is:

```
"2

;; Inspired by: Kenneth R. Baker, Introduction to Sequencing and Scheduling,
;;          New York 1974, p. 95: "Sequence-dependent Setup Times".

Net 6

Restrict 1 2 Started-By
Restrict 1 6 Finished-By

Restrict 2 3 Before Meets
Restrict 2 4 Before Meets
Restrict 2 5 Before Meets
Restrict 2 6 Before

;; If included, the following edge rules (and others missing) can be applied
;; to reduce search space ...

if 2 3 Meets then 2 4 Before
```

[2]Quotation marks serve here to indicate begin and end of a file describing a temporal situation or a solution thereof.

```
Restrict 3 5 After Before Meets Met-By
Restrict 3 6 Before Meets

Restrict 4 5 After Before Meets Met-By
Restrict 4 6 Before Meets

Restrict 5 6 Before Meets


;; If included, the following edge rules (and others missing) can be applied
;; to reduce search space ...

if 2 3 Meets then 2 4 Before
if 2 3 Meets then 2 5 Before
if 2 4 Meets then 2 3 Before
if 2 4 Meets then 2 5 Before
if 2 5 Meets then 2 3 Before
if 2 5 Meets then 2 4 Before


;; Matrix of setup times (coded as if-then rules). Notice that order 1 is called "order 2" here, etc.

if 2 3 Meets then 2 Duration 30 30
if 2 4 Meets then 2 Duration 50 50
if 2 5 Meets then 2 Duration 90 90

if 2 3 Met-By then 3 Duration 40 40
if 3 4 Meets then 3 Duration 20 20
if 3 5 Meets then 3 Duration 80 80
if 3 6 Meets then 3 Duration 40 40

if 2 4 Met-By then 4 Duration 30 30
if 3 4 Met-By then 4 Duration 30 30
if 4 5 Meets then 4 Duration 60 60
if 4 6 Meets then 4 Duration 30 30

if 2 5 Met-By then 5 Duration 20 20
if 3 5 Met-By then 5 Duration 15 15
if 4 5 Met-By then 5 Duration 10 10
if 5 6 Meets then 5 Duration 20 20


Sopo 1 0 0 * * * *

*
```

Result: TIP considers all n! = 3! = 6 possible solutions and finds the optimal manufacturing sequence 1-2-3-4-1 amongst them.

6.2 Example 2: Order Scheduling Using the 'Greedy Algorithm'

Uses the same example as chapter 6.1. However, the solution shall not be found by enumeration of all schedules by order permutation. Instead, the solution shall be found by using a good heuristic. This heuristic is known as the Closest Unvisited City Problem [Baker 1974], or (more generally) as a Greedy Algorithm [Horowitz et al. 1984].

The situation file is identical to the one of chapter 6.1, except that the following lines are added:

```
*
;; Establish CUC (Closest Unvisited City)

LMinimize 2 to 6

Holds 2 Meets 1
Holds 3 Meets 1
Holds 4 Meets 1
Holds 5 Meets 1
Holds 6 Met-By 1
*
```

This way, exactly one solution is found (because the Holds contraints allow only for one solution). By coincidence, it is not only a heuristically good solution, but the optimium solution of chapter 6.1.

6.3 Example 3: A Solution to The 'n Parallel Machines' Problem in Order Scheduling

In [Huber 1990], a severe limitation of Allen's qualitative temporal logic is described: it is not possible to express non-temporal constraints in Allen's temporal logic. Using the Constrain keyword of TIP 2.0, arithmetical constraints may be introduced to enhance the description of a temporal situation. Thereby it is possible to solve the 'n Parallel Machines' Problem (as stated by Huber [Huber 1990]) for arbitrary n.

In our example, we choose $n = 3$. Thus we have 3 identical machines, and two jobs J_1 and J_2 which shall be processed on these machines. Job J_1 shall only be produced on machine 1. Splitting job J_2 is admissible, i. e. this job may be splitted into several jobs which may be processed on different machines.

```
*
Net 7

Label 1 "The root interval which contains all other intervals."
Label 2 "Job 1 on machine 1. This job shall always be produced on machine 1."
Label 3 "Job 2 on machine 1."
Label 4 "Job 2 on machine 2."
Label 5 "Job 2 on machine 3."
Label 6 "Auxiliary interval whose duration equals the sum of durations of intervals 3 and 4."
Label 7 "Auxiliary interval whose duration equals the sum of duration of intervals 5 and 6."

Restrict 1 2 Started-By
Restrict 1 3 Finished-By Contains
Restrict 1 4 Started-By Equals
Restrict 1 5 Started-By Equals

Restrict 2 3 Meets
Restrict 2 4 Starts Started-By Equals
Restrict 2 5 Starts Started-By Equals

Restrict 3 4 After Overlapped-By Met-By Finishes During
Restrict 3 5 After Overlapped-By Met-By Finishes During
Restrict 4 5 Equals

Sopo 1 0 0 * 300 * *
Sopo 2 0 0 100 100 100 100
Sopo 3 100 100 * * 0 500
Sopo 4 0 0 * * 0 500
Sopo 5 0 0 * * 0 500

Sopo 6 * * * * * *
Sopo 7 * * * * 799 801

Constrain 3 4 6
Constrain 5 6 7
"
```

TIP 2.0 finds the two possible solutions which hold under the given constraints:

```
"
13 is number of simple relations

6 is current qualitative solution ...
1 is current quantitative solution ...

          1           1           2 STARTED-BY
          2           1           3 CONTAINS
          3           1           4 EQUALS
          4           1           5 EQUALS
          7           2           3 MEETS
          8           2           4 STARTS
          9           2           5 STARTS
         12           3           4 DURING
         13           3           5 DURING
         16           4           5 EQUALS

 Sopos:

      1          0          0        300        300        300        300
        "The root interval which contains all other intervals."
      2          0          0        100        100        100        100
        "Job 1 on machine 1. ..."
      3        100        100        299        299        199        199
        "Job 2 on machine 1."
      4          0          0        300        300        300        300
        "Job 2 on machine 2."
      5          0          0        300        300        300        300
        "Job 2 on machine 3."
      6          0      15884        499      16383        499        499
        "... equals the sum of durations of intervals 3 and 4."
      7          0      15584        799      16383        799        799
        "... equals the sum of durations of intervals 5 and 6."


10 is current qualitative solution ...
2 is current quantitative solution ...

          1           1           2 STARTED-BY
          2           1           3 FINISHED-BY
          3           1           4 EQUALS
          4           1           5 EQUALS
          7           2           3 MEETS
          8           2           4 STARTS
          9           2           5 STARTS
         12           3           4 FINISHES
         13           3           5 FINISHES
         16           4           5 EQUALS
```

```
Sopos:

    1        0        0      299     300      299      300
      "The root interval which contains all other intervals."
    2        0        0      100     100      100      100
      "Job 1 on machine 1. ..."
    3      100      100      299     300      199      200
      "Job 2 on machine 1."
    4        0        0      299     300      299      300
      "Job 2 on machine 2."
    5        0        0      299     300      299      300
      "Job 2 on machine 3."
    6        0    15884      499   16383      499      500
      "... equals the sum of durations of intervals 3 and 4."
    7        0    15584      799   16383      799      800
      "... equals the sum of durations of intervals 5 and 6."
".
```

6.4 Example 4: A Real Life Example from the Building Area

In [Van Hentenryck 1989] and originally in [Bartusch 1983], a real life planning problem from the building industry (figure 2) is used for benchmarking different types of programs devoted to planning using the Critical Path Method (CPM). A situation described using TIP (Appendix A) finds a solution in a few seconds, which otherwise can be represented using two charts (figures 3 and 4).

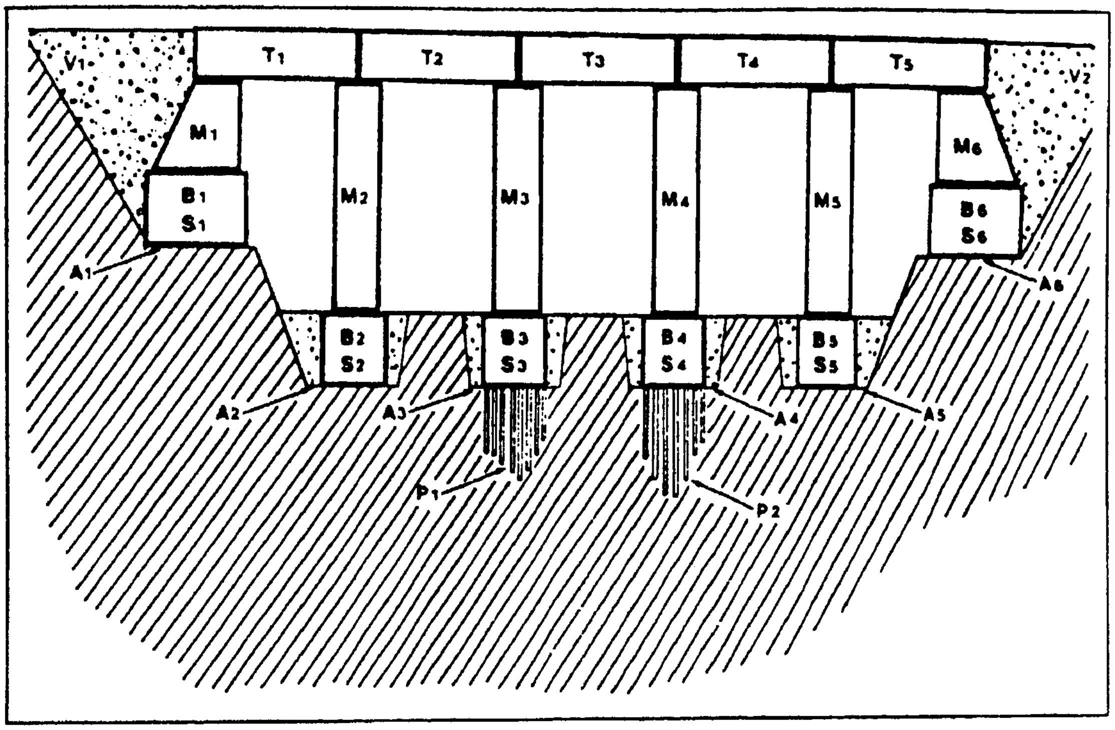


Figure 2: Drawing for example of building area (five pillar bridge) [Bartusch 1983]

N	Name	description	duration	resource
1	PA	beginning of project	0	-
2	A1	excavation (abutment 1)	4	excavator
3	A2	excavation (pillar 1)	2	excavator
4	A3	excavation (pillar 2)	2	excavator
5	A4	excavation (pillar 3)	2	excavator
6	A5	excavation (pillar 4)	2	excavator
7	A6	excavation (pillar 5)	5	excavator
8	P1	foundation piles 2	20	pile-driver
9	P2	foundation piles 3	13	pile-driver
10	UE	erection of temporary housing	10	-
11	S1	formwork (abutment 1)	8	carpentry
12	S2	formwork (pillar 1)	4	carpentry
13	S3	formwork (pillar 2)	4	carpentry
14	S4	formwork (pillar 3)	4	carpentry
15	S5	formwork (pillar 4)	4	carpentry
16	S6	formwork (abutment 2)	10	carpentry
17	B1	concrete foundation (abutment 1)	1	concrete-mixer
18	B2	concrete foundation (pillar 1)	1	concrete-mixer
19	B3	concrete foundation (pillar 2)	1	concrete-mixer
20	B4	concrete foundation (pillar 3)	1	concrete-mixer
21	B5	concrete foundation (pillar 4)	1	concrete-mixer
22	B6	concrete foundation (abutment 2)	1	concrete-mixer
23	AB1	concrete setting time (abutment 1)	1	-
24	AB2	concrete setting time (pillar 1)	1	-
25	AB3	concrete setting time (pillar 2)	1	-
26	AB4	concrete setting time (pillar 3)	1	-
27	AB5	concrete setting time (pillar 4)	1	-
28	AB6	concrete setting time (abutment 2)	1	-
29	M1	masonry work (abutment 1)	16	bricklaying
30	M2	masonry work (pillar 1)	8	bricklaying
31	M3	masonry work (pillar 2)	8	bricklaying
32	M4	masonry work (pillar 3)	8	bricklaying
33	M5	masonry work (pillar 4)	8	bricklaying
34	M6	masonry work (abutment 2)	20	bricklaying
35	L	delivery of the preformed bearers	2	crane
36	T1	positioning (preformed bearer 1)	12	crane
37	T2	positioning (preformed bearer 2)	12	crane
38	T3	positioning (preformed bearer 3)	12	crane
39	T4	positioning (preformed bearer 4)	12	crane
40	T5	positioning (preformed bearer 5)	12	crane
41	UA	removal of the temporary housing	10	-
42	V1	filling 1	15	Caterpillar
43	V2	filling 2	10	Caterpillar
44	K1	costing point 1	0	-
45	K2	costing point 2	0	-
46	PE	end of project	0	-

Figure 3: Data for example from building area [van Hentenryck 1989]

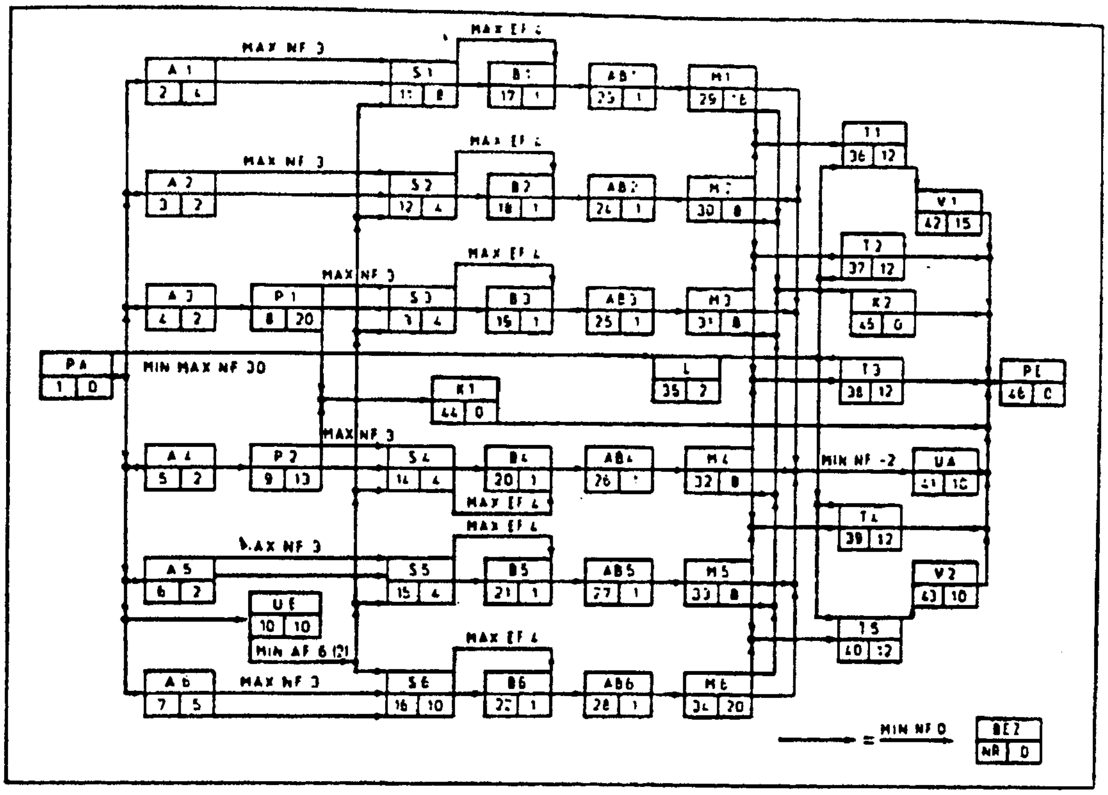


Figure 4: CPM Plan for example from building area [Bartusch 1983]

6.5 Example 5: Representing an AIPLANNER Order Schedule

AIPLANNER is a production planning expert system for job shops. (Cf. other tasks within ESPRIT 2434.) Following chapter 6.1, an AIPLANNER plan can be readily translated into a TIP-based representation. An auxiliary program is available which converts a valid AIPLANNER plan to a situation describing the same plan using TIP 2.0. The auxiliary program creates a string of orders, setup times and break times, which unfold along the time line using the primitive relations Meets and Met-By.

An original plan as made available for the foreman in a manufacturing line:

```
"
Workstation    Auftrag              Anz.   Start        Ende

AX2
               A1.PR-A.E9013         500   9012/ 6:00   9012/ 7:23
               A2.PR-B.E9013         500   9012/ 7:23   9012/ 8:47
               Umrüstzeit                  9012/ 8:47   9012/ 9:07
               A3.PR-D.E9013        1000   9012/ 9:07   9012/17:27
               A5.PR-D.E9014        1000   9012/17:27   9013/ 9:47
               Umrüstzeit                  9013/ 9:47   9013/10:07
               A4.PR-A.E9014         500   9013/10:07   9013/11:30
SMD1
               A1.PR-A.E9013         500   9012/ 7:24   9012/ 8:48
               A2.PR-B.E9013         500   9012/ 8:48   9012/10:11
               Umrüstzeit                  9012/15:28   9012/17:28
               A3.PR-D.E9013        1000   9012/17:28   9012/20:14
               A5.PR-D.E9014        1000   9013/ 9:48   9013/12:34
```

```
              Umrüstzeit                                9013/12:34    9013/14:34
              A4.PR-A.E9014                500          9013/14:34    9013/15:58
"
```

The same plan translated to a situation described using the TIP language:

```
Net 15

Label 1 "Root intervals. Contains all orders on all machines."

Label 2 "Order 1 on machine AX2."
Label 3 "Order 2 on machine AX2."
Label 4 "Setup time between orders 2 and 3 on machine AX2."
Label 5 " Order 3 on machine AX2."
Label 6 "Order 5 on machine AX2."
Label 7 "Setup time between orders 5 and 4 on machine AX2."
Label 8 "Order 4 on machine AX2."

Label 9 "Order 1 on machine SMD1."
Label 10 "Order 2 on machine SMD1."
Label 11 "Setup time between orders 2 and 3 on machine SMD1."
Label 12 " Order 3 on machine SMD1."
Label 13 "Order 5 on machine SMD1."
Label 14 "Setup time between orders 5 and 4 on machine SMD1."
Label 15 "Order 4 on machine SMD1."

Restrict 1 2 Started-By
Restrict 1 15 Finished-By

Restrict 2 3 Meets
Restrict 3 4 Meets
Restrict 4 5 Meets
Restrict 5 6 Meets
Restrict 6 7 Meets
Restrict 7 8 Meets

Restrict 9 10 Meets
Restrict 10 11 Before
Restrict 11 12 Meets
Restrict 12 13 Before
Restrict 13 14 Meets
Restrict 14 15 Meets

Restrict 2 9 Before
Restrict 3 10 Before
Restrict 5 12 Before
Restrict 6 13 Before
Restrict 8 15 Before

;; Resolution: minutes. Time scale starts at week 1 of year.

Sopo 1 * * * * * *

Sopo 2 1800 1883
Sopo 3 1883 1967
Sopo 4 1967 1987
Sopo 5 1987 2487
Sopo 6 2487 3467
Sopo 7 3467 3487
Sopo 8 3487 3570

Sopo 9 1884 1968
Sopo 10 1968 2051
Sopo 11 2368 2488
Sopo 12 2488 2654
Sopo 13 3468 3634
Sopo 14 3634 3754
Sopo 15 3754 3838
*.
```

7. Outlook (Proposed Enhancements of TIP)

7. 1 General Outlook

The TIP Language 2.0, as specified in this language description, is a complete and efficient tool for the description of temporal situations as described by Allen et al. As is indicated by the inclusion of certain procedural, arithmetic and integer programming enhancements, the TIP language might not be complete yet; many more such enhancements may be added.

Furthermore, we would like to add

- a capability for performing branch-and-bound algorithms;

- a capability for also using quantitative constraints to immediately pruning search space. As the TIP inference engine is now, quantitative constraints are only applied after a consistent qualitative solutions was found.

By the incorporation of such features, TIP might resemble more and more general purpose constraint propagation languages such as the CHIP language described in [Van Hentenryck 1989]. However, TIP is very strong in the domain of efficient time representation (by the internal design of its temporal inference engine; and possibly by the use of special hardware accelerators [Jakob 1991]), and representation of uncertainty in the time domain. In these domains, it cannot be paralleled by general purpose constraint propagation languages, which always cope with inefficient internal data representations *because of their generality*.

Future will show how successful TIP can be applied to its intended application domain (planning and scheduling tasks from the CIM domain). These application will in turn mature the TIP language by incorporation of new language features when needed.

7.2 Proposed TIP Enhancements

List of proposed enhancements:

- A new TIP-internal representation of sopos which allows for monotic reasoning (truth maintenance) of sopos.

- An improved TIP-internal representation of "triangles" (three-consistency of edges as described by Allen [Allen 1983]), where an index of triangles is constantly updated (and not only when submitting the keywords Propagate or Simplify to the TIP interpretere).

- A choice of search strategies when making all edges of a scope unique. (Currently, there is only one built-in depth-first strategy available). This necessitates the introduction of a stack of "choicepoints".

8. References

[Allen 1983] J. F. Allen, Maintaining Knowledge About Temporal Intervals; in: Comm. ACM, Vol. 26 No. 11, November 1983, pp. 832 - 843.

[Baker 1974] K. R. Baker, Introduction to Sequencing and Scheduling, New York 1974 (John Wiley Interscience).

[Bartusch 1983] M. Bartusch, Optimierung von Netzplänen mit Anordnungsbeziehungen bei knappen Betriebsmitteln, Dissertation an der Fakultät für Bauwesen der Rheinisch-Westfälischen Technischen Hochschule Aachen 1983, Aachen 1983.

[Becker 1988] S. U. Becker, Konzeption und Implementation einer Temporalen Inferenzkomponente -- Anwendung auf Planungsprobleme in der Fließproduktion, Diplomarbeit am Fachbereich Informatik der Universität Hamburg 1988, Hamburg 1988.

[Horowitz et al. 1984] Horowitz, E., Sahni, S., Fundamentals of Computer Algorithms, Rockville (USA) 1984 (Computer Science Press).

[Huber 1990] A. Huber, Dynamic Scheduling under Different Production Strategies; in: ESPRIT 2434 Consortium / Philips GmbH [ed.], ESPRIT 2434 18 Months Report, Supplement B, Task 2.5, Hamburg 1990.

[Jakob 1991] Temporal Inference Unit -- Implementing An AI Coprocessor; in: ESPRIT 2434 Consortium / Philips GmbH [ed.], ESPRIT 2434 24 Months Report, Task 2.5, Hamburg 1991.

[Rit 1986] J. - F. Rit, Propagating Temporal Constraints for Scheduling, Proc. Conf. American Assoc. for Artificial Intelligence 186, pp. 383 - 388.

[Van Hentenryck 1989] P. v. Hentenryck, Constraint Satisfaction in Logic Programming, Cambridge (Massachusetts / USA) 1989 (The MIT Press).

[Wirth 1974] N. Wirth, Algorithms + Data Structures = Programs, Englewood Cliffs (New Jersey / USA) 1976 (Prentice-Hall).

Appendix A: Situation File of Example 4

```
;; Inspired by: M. Bartusch, Optimierung von Netzplänen mit Anordnungs-
;; beziehungen bei knappen Betriebsmitteln, Aachen 1983; p. 134.

;; assumes that PointP = TRUE


Net 63

;; Sopos 1 to 46 map directly onto activities ('Vorgänge') 1 to 46 of CPM plan.
;; Sopos 47 to 63 are auxiliary sopos to implement the CPM constraints (EF, NF, AF, SF).

Relax 1 to 63

;; durations by 'Vorgang'
Sopo 1 * * * * 0 0
Sopo 2 * * * * 400 400
Sopo 3 * * * * 200 200
Sopo 4 * * * * 200 200
Sopo 5 * * * * 200 200
Sopo 6 * * * * 200 200
Sopo 7 * * * * 500 500
Sopo 8 * * * * 2000 2000
Sopo 9 * * * * 1300 1300
Sopo 10 * * * * 1000 1000
Sopo 11 * * * * 800 800
Sopo 12 * * * * 400 400
Sopo 13 * * * * 400 400
Sopo 14 * * * * 400 400
Sopo 15 * * * * 400 400
Sopo 16 * * * * 1000 1000
Sopo 17 * * * * 100 100
Sopo 18 * * * * 100 100
Sopo 19 * * * * 100 100
Sopo 20 * * * * 100 100
Sopo 21 * * * * 100 100
Sopo 22 * * * * 100 100
Sopo 23 * * * * 100 100
Sopo 24 * * * * 100 100
Sopo 25 * * * * 100 100
Sopo 26 * * * * 100 100
Sopo 27 * * * * 100 100
Sopo 28 * * * * 100 100
Sopo 29 * * * * 1600 1600
Sopo 30 * * * * 800 800
Sopo 31 * * * * 800 800
Sopo 32 * * * * 800 800
Sopo 33 * * * * 800 800
Sopo 34 * * * * 2000 2000
Sopo 35 * * * * 200 200
Sopo 36 * * * * 1200 1200
Sopo 37 * * * * 1200 1200
Sopo 38 * * * * 1200 1200
Sopo 39 * * * * 1200 1200
Sopo 40 * * * * 1200 1200
Sopo 41 * * * * 1000 1000
Sopo 42 * * * * 1500 1500
Sopo 43 * * * * 1000 1000
Sopo 44 * * * * 0 0
Sopo 45 * * * * 0 0
Sopo 46 * * * * 0 0

;; sequences by 'Vorgang'
Restrict 1 2 Before
Restrict 2 11 Before
Restrict 2 48 Meets
Restrict 11 48 Met-By
Restrict 11 17 Before
Restrict 11 49 Meets
Restrict 17 49 Finishes
Restrict 17 23 Before
Restrict 23 29 Before

Restrict 1 3 Before
Restrict 3 12 Before
Restrict 3 50 Meets
Restrict 12 50 Met-By
Restrict 12 18 Before
Restrict 12 51 Meets
Restrict 18 51 Finishes
Restrict 18 24 Before
Restrict 24 30 Before

Restrict 1 4 Before
Restrict 4 8 Before
Restrict 8 13 Before
Restrict 8 52 Meets
Restrict 13 52 Met-By
Restrict 13 19 Before
Restrict 13 62 Meets
Restrict 19 62 Finishes
Restrict 19 25 Before
Restrict 25 31 Before

Restrict 1 5 Before
Restrict 5 9 Before
Restrict 9 14 Before
Restrict 9 53 Meets
Restrict 14 53 Met-By
Restrict 14 20 Before
Restrict 14 54 Meets
Restrict 20 54 Finishes
Restrict 20 26 Before
Restrict 26 32 Before

Restrict 1 6 Before
Restrict 6 15 Before
Restrict 6 55 Meets
Restrict 15 55 Met-By
Restrict 15 21 Before
Restrict 15 56 Meets
Restrict 21 56 Finishes
Restrict 21 27 Before
Restrict 27 33 Before
```

Restrict 1 7 Before
Restrict 7 16 Before
Restrict 7 57 Meets
Restrict 16 57 Met-By
Restrict 16 22 Before
Restrict 16 58 Meets
Restrict 22 58 Finishes
Restrict 22 28 Before
Restrict 28 34 Before

Restrict 8 44 Before
Restrict 9 44 Before
Restrict 1 10 Meets
Restrict 10 59 Starts Started-By Equals
Restrict 11 59 After
Restrict 12 59 After
Restrict 13 59 After
Restrict 14 59 After
Restrict 15 59 After
Restrict 16 59 After

Restrict 29 36 Before
Restrict 30 36 Before
Restrict 30 37 Before
Restrict 31 37 Before
Restrict 31 38 Before
Restrict 32 38 Before
Restrict 32 39 Before
Restrict 33 39 Before
Restrict 33 40 Before
Restrict 34 40 Before

Restrict 29 45 Before
Restrict 30 45 Before
Restrict 31 45 Before
Restrict 32 45 Before
Restrict 33 45 Before
Restrict 34 45 Before

Restrict 29 60 Before
Restrict 30 60 Before
Restrict 31 60 Before
Restrict 32 60 Before
Restrict 33 60 Before
Restrict 34 60 Before
Restrict 60 63 Met-By
Restrict 41 63 Started-By

Restrict 1 61 Meets
Restrict 35 61 Met-By
Restrict 35 36 Before
Restrict 35 37 Before
Restrict 35 38 Before
Restrict 35 39 Before
Restrict 35 40 Before

Restrict 36 42 Before
Restrict 42 46 Before

```
Restrict 37 46 Before
Restrict 38 46 Before
Restrict 41 46 Before
Restrict 39 46 Before
Restrict 40 43 Before
Restrict 43 46 Before
Restrict 45 46 Before
Restrict 44 46 Before

;; total project
Sopo 47 0 0 * * * *
Restrict 1 47 Starts
Restrict 46 47 Finishes

;; EF, NF, AF, SF constraints
Sopo 48 * * * * 0 300
Sopo 49 * * * * 0 400
Sopo 50 * * * * 0 300
Sopo 51 * * * * 0 400
Sopo 52 * * * * 0 300
Sopo 53 * * * * 0 300
Sopo 54 * * * * 0 400
Sopo 55 * * * * 0 300
Sopo 56 * * * * 0 400
Sopo 57 * * * * 0 300
Sopo 58 * * * * 0 400
Sopo 59 * * * * 600 *
Sopo 60 * * * * 0 0
Sopo 61 * * * * 3000 3000
Sopo 62 * * * * 400 400
Sopo 63 * * * * 200 200
*
```