A New Scheduling Model Based on a Description Logic - Comparison with the Model of Manne

Pok-Son Kim, Arne Kutzner
Kookmin University, Department of Mathematics, Seoul 136-702, Korea
pskim@kookmin.ac.kr
Seokyeong University, Department of E-Business, Seoul 136-704, Korea
kutzner@skuniv.ac.kr

Abstract

We introduce a new scheduling language called $\mathcal{RSVX}$ that encodes scheduling problems into temporal description logics. Scheduling problems are syntactically represented as descriptions (activity terms) in $\mathcal{RSVX}$. Based on the semantics given for $\mathcal{RSVX}$, a calculus is defined which can transform each activity term A into a semantically equivalent, resolved activity term B, which is a nonredundant disjunction of all active schedules derived from A.
Additionally $\mathcal{RSVX}$ is compared with the conventional model of Manne. This comparison clarifies several specially advantageous characteristics of $\mathcal{RSVX}$.

Introduction

Characteristics of description logics are

- a term language for concepts and other notions,
- a clean denotational semantics
- and specific calculi (like subsumption) based on the semantics.

Based on these characteristics, resource constrained project scheduling problems with variants can be represented and solved as shown in [8]. Scheduling problems are syntactically represented as descriptions (compound terms) in a defined logic-based language. Such descriptions are similar to concepts in description logics [12, 6] which emerged from KL-ONE based, terminological knowledge representation systems of artificial intelligence [12]. The use of a semantic method from description logics is the key for understanding the meaning of compound terms. Further such a description language allows compound terms to be represented by means of diagrams with a scan-line. From such diagrams the flow structure and the content of scheduling problems (for example, what activity requires what resource) can be read easy and directly. The process to find an optimal conflict free schedule for a given schedule problem, i.e. given tasks, their duration and resource usage, can also be described by means of diagrams.

In this article we introduce a new scheduling language called $\mathcal{RSVX}$ that owns characteristics of description logics. For a reduced activity-term in $\mathcal{RSVX}$ there can exist several different active schedules and all non-redundant active schedules can be generated using the algorithm $\mathcal{A}_{\mathcal{RSV}}$. Until now active schedules have been represented by $\mathcal{RSV}$-diagrams. We show active schedules can also be represented as reduced activity terms in $\mathcal{RSVX}$. So a calculus can be defined which transforms each reduced activity-term A into another activity-term A' representing all non-redundant active schedules derived from A. Finally we compare the computation model $\mathcal{RSVX}$ with the conventional model of Manne [10]. With this we show several specially advantageous characteristics of $\mathcal{RSVX}$.

The Scheduling Language $\mathcal{RSVX}$

A terminological language $\mathcal{RSVX}$ that can be used to model a new general class of resource-constrained project scheduling problems is defined as follows:

The syntax of the language $\mathcal{RSVX}$

An expression of $\mathcal{RSVX}$ holds implicitly the following general constraints:

- Each atomic (ground) activity i is associated with a resource $r(i)$ and an activity time $d(i)$. A resource (machine) and an activity time are needed for completing the atomic activity.
- Except for a dummy resource eu each resource can be assigned to only one activity at a time; a dummy resource eu is unlimitedly available (resource constraint).
- Activity splitting is not allowed (non-preemptive case).

Definition 0.1. *The vocabulary of $\mathcal{RSVX}$ consists of two disjoint sets of symbols. These sets are:*

- *A set of atomic activities* $\{(0, eu, i)\} \cup \{(i, r(i), d(i)) | i = 1, \cdots, n, r(i) \in R, d(i) \in \mathbb{N}\}$ *where* $(0, eu, i)$ *corresponds a dummy atomic activity and* R *is a finite set of resources.*
- *A set of structural symbols (operators)* '**seq**', '**pll**' *and* '**xor**'. *The structural symbol* '**xor**' *is called* designated *operator.*

First the reduced activity-terms (r-activity-term for short) *of* $\mathcal{RSVX}$ *are given inductively as follows:*

1. *Each atomic activity is a r-activity-term.*
2. *If* $A_1, A_2, \cdots, A_k$ *are r-activity-terms, then*

$$(\mathbf{seq}\, A_1, A_2, \cdots, A_k) \text{ and}$$
$$(\mathbf{pll}\, A_1, A_2, \cdots, A_k)$$

are r-activity-terms.

Now, activity-terms *of* $\mathcal{RSVX}$ *are formed as follows :*

Each r-activity-term is an activity-term. If $B_1, B_2, \cdots, B_k$ *are r-activity-terms, then*

$$(\mathbf{xor}\, B_1, B_2, \cdots, B_k)$$

is an activity-term.

The operators '**seq**' and '**pll**' have the following meaning:

- '**seq**' : This operator specifies the *sequential* processing of an r-activity-term or r-activity-terms (precedence constraints).
- '**pll**' : This operator specifies the possibility of parallel processing of r-activity-terms.

The designated operator '**xor**' has the following meaning:

- '**xor**' : This operator can be used to combine several r-activity-terms one of which may be alternatively chosen.

Schedules

We take the definition of active schedules from [8]. Let A be a r-activity-term and let $A_1, \cdots, A_n$ be all atomic activities occurring in A. An *active schedule* for A is a set of starting times of atomic activities $\{t_{A_i} \in \mathbb{N} \,|\, A_i, i = 1, 2, \cdots, n\}$ such that:

- The precedence constraints are satisfied,
- The resource constraints are satisfied and
- No atomic activity can be started earlier without changing other start times.

The *makespan* of an active schedule is the duration from the first starting time $\min_i(t_{A_i})$ to the stopping time $\max_i(t_{A_i} + d(A_i))$.

The semantics of the language $\mathcal{RSVX}$

Definition 0.2. *The model-theoretic semantics of activity-terms in* $\mathcal{RSVX}$ *is given by an interpretation* $\mathcal{I}$ *which consists of the set* $\mathcal{D}$ *(the domain of* $\mathcal{I}$*) and a function* $\cdot^{\mathcal{I}}$ *(the interpretation function of* $\mathcal{I}$*). The set* $\mathcal{D}$ *consists of all active schedules derived from activity-terms in* $\mathcal{RSVX}$*. The interpretation function* $\cdot^{\mathcal{I}}$ *assigns to every activity-term* A *some subset of* $\mathcal{D}$ *that consists of all active schedules derived from* A*.*

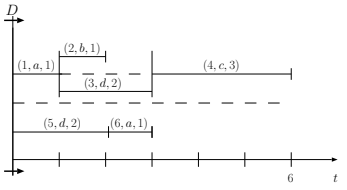
Solving the $\mathcal{RSVX}$-problem using diagram-based calculation

Many varieties of branch-and-bound-based implicit enumeration methods ([13], [14], [15], [1], [4], [5], [11], [2]) for solving the resource constrained scheduling problem which may also be used for determining the optimal schedules for each r-activity term in $\mathcal{RSVX}$ have been reported. Further using the *new* diagram and scan-line-based algorithm $\mathcal{A}_{\mathcal{RSV}}$ presented by Kim and Schmidt-Schauß [8] each r-activity-term in $\mathcal{RSVX}$ can be represented graphically and explicit generation of all nonredundant active schedules for any r-activity-term can be illustrated graphically. One objective of this article is to show that with the aid of the diagram representation used in $\mathcal{A}_{\mathcal{RSV}}$ each active schedule can also be represented as a r-activity term in $\mathcal{RSVX}$.

Solution algorithm $\mathcal{A}_{\mathcal{RSV}}$ based on a scan-line principle

To demonstrate the computation process of $\mathcal{A}_{\mathcal{RSV}}$ we consider the following activity (below, we refer to it as A) and its $\mathcal{RSV}$-diagram representation:

$$\mathbf{pll}\quad (\mathbf{seq}\,(1, a, 1), (\mathbf{pll}(2, b, 1), (3, d, 2)), (4, c, 3)),$$
$$(\mathbf{seq}\,(5, d, 2), (6, a, 1))$$



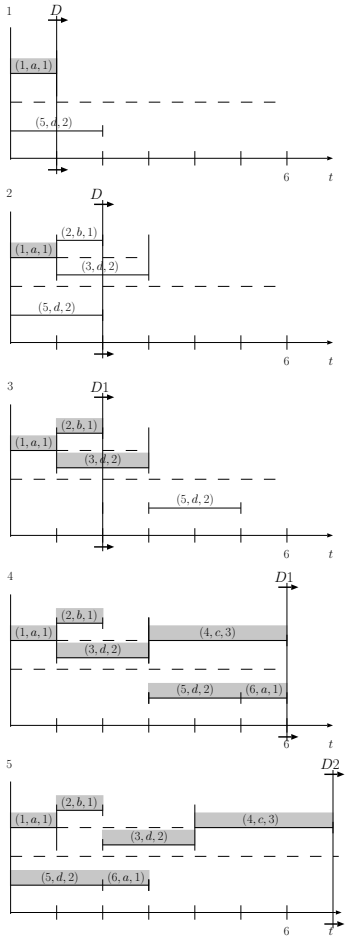


Figure 1: $\mathcal{RSV}$-diagram-based calculation

In the beginning the scan-line is found at the time $t_{SL} = 0$ and the diagram is empty. In the diagram each atomic activity i has a left and a right end point (a start and end time), denoted by $LE(i)$ and $RE(i)$ and they are referred to as the *stopping times* of the scan-line. (D, t) with $t \geq 0$ denotes the scan-line is found at the stopping time $t_{SL} = t$ in the diagram D. Instead of a continuous moving, the scan-line jumps from a stopping time into the next stopping time while determining and then resolving resource conflicts.

Step 1: **Attaching start atomic activities to the scan-line:** First all start atomic activities of A, 1 and 5, which have no predecessors in A are attached to the scan-line. "Attaching an atomic activity i to the scan-line" means that i is placed in the diagram so that the time at which the scan-line is found is assigned to i as its start time.

Step 2: **Moving the scan-line:** The scan-line jumps to the next stopping time $t_{SL} = 1$.

Step 3: **Determining and resolving resource conflicts (Multiplying the diagram by the number of the existing conflict combinations); Freezing all definitely placed atomic activities:** First, because the begin and end times of conflict-free activities i with $RE(i) = t_{SL}$ have been definitely determined, all such conflict-free activities are frozen in order to mark that they must not be moved any more. If several scan-line activities (activities i with $LE(i) < t_{SL}$ and $RE(i) \geq t_{SL}$) require a conflict resource r simultaneously, a resource conflict occurs. A resource conflict is resolved while selecting only one activity and all the other activities are moved behind the selected activity. In this case the begin and end time of this selected activity are definitely determined. In order to mark this it is frozen.
At any stopping time t_{SL}, several different t_{SL}-time resource conflicts can simultaneously occur. In this case exactly one (t_{SL}, r)-conflict activity for *each* t_{SL}-time conflict resource r is selected in order to freeze them. There exist several different combinations for selecting activities. Such a combination is called *a conflict combination.* For the above example, we have diagram 1 of figure 1 where $t_{SL} = 1$ has no resource conflict and the 1-time conflict-free activity 1 has been frozen.

Step 4: **Deleting all t_{SL}-time direct scan-line activities from the activity-term A:** In (D, t_{SL}) all t_{SL}-time direct scan-line activities i.e. activities i with $RE(i) = t_{SL}$have been surely frozen in the last step 3. Now they are deleted from the input activity A. So A may become smaller. For our example the 1-time direct scan-line activity 1 is deleted from the actual activity term A.

Step 5: **Attaching further atomic activities to the scan-line:** Further atomic activities from the actual term A which can be attached to the scan-line are determined in order to place them. If in (D, t_{SL}) a scan-line activity i with $RE(i) > t_{SL}$ has been frozen, the resource r of i is being blocked until the time $RE(i)$. So, all further atomic activities requiring *the t_{SL}-time blocked resource* r which have not yet been placed in the diagram and have no predecessor in A must wait until the scan-line has jumped to the time $RE(i)$. The other activities of A having no predecessor and having not yet been placed are attached to the scan-line. For the above example A has two atomic activities 2 and 3 which have no predecessor and the resources required by 2 and 3 are not blocked. So 2 and 3 are attached to the scan-line.

Furthermore the steps 2, 3, 4 and 5 are recursively applied until all atomic activities have been placed in the diagram and all activities in the diagram have been frozen so that A is empty and an active schedule is completed. For the example, the scan-line jumps into the next stopping time $t_{SL} = 2$ and we have the resulting diagram 2 of figure 1, in which 3 and 5 give rise to a resource conflict. So there are 2 2-time conflict combinations [3] and [5]. The diagram is duplicated, let these be $D1$, $D2$ and [3] and [5] are assigned to $D1$ and $D2$ respectively. In every diagram, the 2-time conflict free activity 2 and the assigned 2-time conflict activity are frozen and the other 2-time conflict activity is moved behind each corresponding frozen activity. Subsequently we proceed with

the next step 4 in every diagram.
If we pursue $(D1, 2)$ to which the combination [3] is assigned, we have diagram 3 of figure 1 where 5 has been moved behind 3. Now the two 2-time direct scan-line activity 2 in $(D1, 2)$ (diagram 3 of figure 1) has to be deleted from the term A. After deleting the activity we have the following *new* A for $(D1, 2)$:

$$\begin{array}{ll} \textbf{pll} & (\textbf{seq}\,(3, d, 2), (4, c, 3)), \\ & (\textbf{seq}\,(5, d, 2), (6, a, 1)) \end{array}$$

3 and 5 have no predecessor but they are already located in the diagram. So in the next step 5 there is no activity to be attached to the scan-line. After applying the further steps 2, 3, 4 and 5 to the diagram $(D1, 2)$ recursively, we have diagram 4 of figure 1 (an active schedule). From the diagram $(D2, 2)$, another active schedule with project makespan 6 (diagram 5 of figure 1) is generated.

Representation of active schedules as r-activity-terms in $\mathcal{RSVX}$

For a r-activity-term there can exist several active schedules requiring different project makespan. Until now active schedules were represented by $\mathcal{RSV}$-diagrams or Gantt-charts. In this section it is shown that active schedules can also be described as r-activity terms in $\mathcal{RSVX}$.
To resolve resource conflicts the atomic activities carried out first are frozen and all the other conflict atomic activities are moved to right behind the frozen activities. Occupying each arising empty interval from the movement by dummy-atomic activities, the resulting diagrams can be represented by r-activity-terms. So, by this way, each active schedule can also be represented by the corresponding r-activity term. The active schedules 4 and 5 of figure 1 can be represented by the following r-activity terms in $\mathcal{RSVX}$:

$$\begin{array}{l} \textbf{pll}(\textbf{seq}\,(1, a, 1), (\textbf{pll}(2, b, 1), (3, d, 2)), (4, c, 3)), \\ \quad (\textbf{seq}\,(0, eu\,, 3), (5, d, 2), (6, a, 1)) \end{array}$$

$$\begin{array}{l} \textbf{pll}(\textbf{seq}(1, a, 1), (\textbf{pll}(2, b, 1), (\textbf{seq}(0, eu\,, 1), (3, d, 2))), (4, c, 3)), \\ \quad (\textbf{seq}\,(5, d, 2), (6, a, 1)) \end{array}$$

The $\mathcal{RSVX}$-calculus

Given a r-activity term H in $\mathcal{RSVX}$ as input to the algorithm $\mathcal{A}_{\mathcal{RSV}}$, $\mathcal{A}_{\mathcal{RSV}}$ generates all active schedules derivable from H. Let $H_1, \cdots, H_k$ be r-activity terms representing all these active schedules derived from H. Then the expression $(\textbf{xor}\, H, \cdots, H_k)$ is also an activity term in $\mathcal{RSVX}$. We call such a term *a resolved activity term*. So we can define a calculus which transforms a r-activity term into a resolved activity term in $\mathcal{RSVX}$.

Definition 0.3. ***Transformation into resolved activity term***

A transformation rule called $\mathcal{RSVX}$-calculus is defined as follows:

By means of the algorithm $\mathcal{A}_{\mathcal{RSV}}$ each r-activity term H can be transformed into a resolved activity term H_r, denoted by $H \rightarrow_{\mathcal{A}_{\mathcal{RSV}}} H_r$.

Theorem 0.1. ***Soundness of $\mathcal{RSVX}$-calculus***

The $\mathcal{RSVX}$-calculus is correct (sound).

Proof. It is obvious that $H \rightarrow_{\mathcal{A}_{\mathcal{RSV}}} H_r$ implies $H \models H_r$, i. e. $(H)^{\mathcal{I}} = (H_r)^{\mathcal{I}}$.

A Comparison of Solving Methods

In this section the computation model $\mathcal{RSVX}$ with $\mathcal{A}_{\mathcal{RSV}}$ is compared to a conventional model. This comparison clarifies several specially advantageous characteristics of the model $\mathcal{RSVX}$ with $\mathcal{A}_{\mathcal{RSV}}$.

The model of Manne

Allan S. Manne [10] presented a model for describing and solving the general job shop-problem on the basis of the linear programming. In this section we describe the job shop-problem based on the model of Manne. The represented version of the solution of job shop-problems with the model of Manne is adapted to Fuerst [7].

In a job shop-problem n jobs have to be processed on m machines assuming the following facts [3]:

- A machine can process only one job at a time.
- The processing of a job on a machine is called an operation.
- An operation cannot be interrupted.
- A job consists of at most m operations; a job is processed on a machine at most once.
- The machine sequence of a job is given.
- The operation sequence on the machines are unknown and have to be determined in order to minimize makespan.

In order to describe a job shop-problem, we introduce the following symbols:

I	=	$\{i \mid i = 1, \cdots, n,\ i$ is a job.$\}$
J	=	$\{j \mid j = 1, \cdots, m,\ j$ is a machine.$\}$
Π	=	$\{\pi_i \mid \pi_i = (f_i(1), f_i(2), \cdots, f_i(i_l))$ is the required machine sequence of job i, $i_l (\leq m)$ is the last operation of job i and $f_i(i_l)$ is the corresponding machine.$\}$
d_{ij}		the processing time of job i on machine j
M		a large number

		j			
		1	2	3	4
	1			4	2
i	2	3			3
	3		2	1	2
	4	4	1	2	

Table 1: The duration d_{ij}

variables:

D = makespan

h_{ij} = beginning time of job i on machine j, i.e. beginning time of the operation ij

$$u_{ikj} = \begin{cases} 1, & \text{if job } i \text{ is processed before job } k \\ & \text{on machine } j \\ 0, & \text{otherwise} \end{cases}$$

The goal function is:

$$D = min!$$

The additional conditions to be satisfied are:

1. $h_{ij} + d_{ij} \leq h_{ij'}$ for all $i \in I$; for all $j = f_i(r)$ and $j' = f_i(r+1)$ with $r = 1, \cdots, i_l - 1$
2. $h_{if_i(i_l)} + d_{if_i(i_l)} \leq D$ for all $i \in I$
3. $h_{ij} + d_{ij} \leq h_{kj} + M \cdot (1 - u_{ikj})$ for all $j \in J$ and $i \neq k$ with $i, k \in \{e | \exists r$, such that $f_e(r) = j$ in $\pi_e\}$
4. $h_{kj} + d_{kj} \leq h_{ij} + M \cdot u_{ikj}$ for all $j \in J$ and $i \neq k$ with $i, k \in \{e | \exists r$, such that $f_e(r) = j$ in $\pi_e\}$
5. $h_{ij} \in \mathbb{N}_0$ for all $i \in I$ and $j \in J$
6. $D \geq 0$
7. $u_{ikj} \in \{0, 1\}$ for all $j \in J$ and $i \neq k$ with $i, k \in \{e | \exists r$, such that $f_e(r) = j$ in $\pi_e\}$

Now we want to compare the two approaches by means of a simple example. First we represent a job shop-problem by means of the model of Manne and then by means of the language $\mathcal{RSVX}$.

Example

Let the following job shop-problem be given: 4 jobs ($i = 1, 2, 3, 4$) are processed on 4 machines ($j = 1, 2, 3, 4$). From table 1 the duration of each operation ij, denoted by d_{ij}, can be read.

Further let the following processing orders π_i of jobs i be given:

$\pi_1 = (4, 3)$
$\pi_2 = (1, 4)$
$\pi_3 = (2, 3, 4)$
$\pi_4 = (3, 1, 2)$

The above problem is formulated according to the model of Manne as follows:

The goal function is:

$$D = min!$$

The additional conditions to be satisfied are:

1. Keeping of the given machine sequence of the jobs:
   job 1: $h_{14} + 2 \leq h_{13}$
   job 2: $h_{21} + 3 \leq h_{24}$
   job 3: $h_{32} + 2 \leq h_{33}$
   $h_{33} + 1 \leq h_{34}$
   job 4: $h_{43} + 2 \leq h_{41}$
   $h_{41} + 4 \leq h_{42}$
2. Determining of the makespan:
   job 1: $h_{13} + 4 \leq D$
   job 2: $h_{24} + 3 \leq D$
   job 3: $h_{34} + 2 \leq D$
   job 4: $h_{42} + 1 \leq D$
3. Decision about the operation sequence on the machines:
   machine 1: $h_{21} + 3 \leq h_{41} + M(1 - u_{241})$
   $h_{41} + 4 \leq h_{21} + M \cdot u_{241}$
   machine 2: $h_{32} + 2 \leq h_{42} + M(1 - u_{342})$
   $h_{42} + 1 \leq h_{32} + M \cdot u_{342}$
   machine 3: $h_{13} + 4 \leq h_{33} + M(1 - u_{133})$
   $h_{33} + 1 \leq h_{13} + M \cdot u_{133}$
   $h_{13} + 4 \leq h_{43} + M(1 - u_{143})$
   $h_{43} + 2 \leq h_{13} + M \cdot u_{143}$
   $h_{33} + 1 \leq h_{43} + M(1 - u_{343})$
   $h_{43} + 2 \leq h_{33} + M \cdot u_{343}$
   machine 4 $h_{14} + 2 \leq h_{24} + M(1 - u_{124})$
   $h_{24} + 3 \leq h_{14} + M \cdot u_{124}$
   $h_{14} + 2 \leq h_{34} + M(1 - u_{134})$
   $h_{34} + 2 \leq h_{14} + M \cdot u_{134}$
   $h_{24} + 3 \leq h_{34} + M(1 - u_{234})$
   $h_{34} + 2 \leq h_{24} + M \cdot u_{234}$
4. Nonnegativity condition:
   $h_{ij} \in \mathbb{N}$ for all relevants i, j
   $D \geq 0$
   $u_{ikj} \in \{0, 1\}$ for all relevants i, k, j

Before we represent this problem as an r-activity-term in $\mathcal{RSVX}$, we identify machines $j = 1, 2, 3, 4$ with the alphabetic characters a, b, c, d respectively. The informations about machine sequence and processing time of each job can be summarized as the following table:

job	machine sequence processing time
1	d c 2 4
2	a d 3 3
3	b c d 2 1 2
4	c a b 2 4 1

The operation ij is denoted by ij. The considered problem can be represented as the following expression in $\mathcal{RSVX}$:

$$\begin{aligned}\textbf{pll}\quad & (\textbf{seq}\,(11,d,2),\,(12,c,4)),\\ & (\textbf{seq}\,(21,a,3),\,(22,d,3)),\\ & (\textbf{seq}\,(31,b,2),\,(32,c,1),\,(33,d,2)),\\ & (\textbf{seq}\,(41,c,2),\,(42,a,4),\,(43,b,1))\end{aligned}$$

Given the above activity-term as input-expression to $\mathcal{A}_{\mathcal{RSV}}$, all active schedules are generated. So the given job shop-problem is solved. The main differences between two above models will be mentioned in the following section.

Conclusion

We introduced a new logic-based scheduling language called $\mathcal{RSVX}$. Scheduling problems are defined as descriptions (activity terms) in $\mathcal{RSVX}$. An active schedule generated by the algorithm $\mathcal{A}_{\mathcal{RSV}}$can also be represented by an activity term in $\mathcal{RSVX}$. Based on this representation method and the semantics given for $\mathcal{RSVX}$ a calculus was defined which can transform each activity term A into a semantically equivalent, resolved activity term B which is a nonredundant disjunction of all active schedules derived from A.
We compared $\mathcal{RSVX}$ with the conventional model of Manne. That comparison clarified the following specially advantageous characteristics of $\mathcal{RSVX}$:

- $\mathcal{RSVX}$ is more powerful than the model of Manne because all problems that can be solved by the model of Manne can also be solved by $\mathcal{RSVX}$-model. However, the inversion is not true.
- Related works (e.g. [9]) often put a value on the models based on the fundamental principles of linear programming. It is argued that such models would give very elegant formulation for job shop-problems. We think $\mathcal{RSVX}$ offers a more elegant and easy understandable formulation for job shop-problems. Further, from a $\mathcal{RSVX}$-expression describing a job shop-problem, the processing structure of operations can be read directly and clear.
- If a job shop problem is described by a $\mathcal{RSVX}$-expression, then the costly representation by numerous inequalities belonging to the model of Manne can be saved.
- The model of Manne comprises the unpleasant property that the number of variables and restrictions (inequalities) increase strongly depending on the problem size. This can be avoided with $\mathcal{RSVX}$.

References

[1] C. E. Bell and K. Park. Solving Resource-Constrained Project Scheduling Problems by A^* Search. *Naval Research Logistics Quarterly*, 37(1):61–84, 1990.

[2] P. Brucker, S. Knust, and O. Schoo, A. Thiele. A Branch and Bound Algorithm for the Resource-constrained Project Scheduling Problem. *European Journal of Operational Research*, 107:272–288, 1998.

[3] J. Carlier and E. Pinson. An Algorithm for Solving the Job-Shop Problem. *Operations Research*, 8:487–503, 1989.

[4] E. Demeulemeester and W. Herroelen. A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem. *Management Science*, 38(12):1803–1818, 1992.

[5] E. Demeulemeester and W. Herroelen. New Benchmark Results for the Resource-constrained Project Scheduling Problem. *Management Science*, 43(11):1485–1492, 1997.

[6] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The Complexity of Concept Languages. *Information and Computation*, 134(1):1–58, 1997.

[7] A. Fürst. *Auftragsfolge- und Personalplanung.* München und Mering, 1997.

[8] P. S. Kim and M. Schmidt-Schauß. A Term-Based Approach to Project Scheduling. *ICCS01, Lecture Notes in Artificial Intelligence Series 2120, p. 304 ff., Springer-Verlag*, 2001.

[9] Klaus-Peter Kistner and Marion Steven. *Produktionsplanung.* Physica-Verlag, Heidelberg, 1990.

[10] A. S. Manne. On the Job-Shop Scheduling Problem. *Operations Research*, 8:219–233, 1960.

[11] A. Mingozzi, V. Maniezzo, and L. Ricciardelli, S. Bianco. An exact Algorithm for Project Scheduling with Resource Constraints based on a New Mathematical Formulation. *Management Science*, 44(5):714–729, 1998.

[12] M. Schmidt-Schauß and G. Smolka. Attributive Concept Descriptions with Unions and Complements. Technical Report SEKI Report SR-88-21, FB Informatik, Universität Kaiserslautern, D-6750, Germany, 1988.

[13] L. Schrage. Solving Resource-Constrained Network Problems by Implicit Enumeration-Nonpreemptive Case. *Operations Research*, 10:263–278, 1970.

[14] J. P. Stinson, E. W. Davis, and B. M. Khumawala. Multiple Resource-Constrained Scheduling Using Branch and Bound. *AIIE Transactions*, 10(3):252–259, 1978.

[15] F. B. Talbot and J. H. Patterson. An Efficient Integer Programming Algorithm with Network Cuts for Solving Resource-Constrained Scheduling Problems. *Management Science*, 24(11):1163–1174, 1978.