A New Term-Based Approach to Project Scheduling
The Scheduling Language $\mathcal{RCPSV}$

Pok-Son Kim

Institut für Wirtschaftsinformatik / Informationsmanagement
Johann Wolfgang Goethe-Universität
Frankfurt am Main, Germany
{kim}@wiwi.uni-frankfurt.de

Abstract We introduce a new logic-based scheduling language called $\mathcal{RCPSV}$ which may be used to model a new general class of resource-constrained project scheduling problems for minimizing the project completion time. Similar to the scheduling language $\mathcal{RSV}$ ([5]), $\mathcal{RCPSV}$ makes scheduling problems possible to be represented as terminological descriptions and permits not only each atomic activity but also each subproject to be accomplished in one of several different ways. We define a calculus for $\mathcal{RCPSV}$-expressions that offers an effective approach for solving the scheduling problem $\mathcal{RCPSV}$. In addition a diagram-based, nonredundant enumeration algorithm similar to that of $\mathcal{RSV}$ is developed.
Though $\mathcal{RSV}$ and $\mathcal{RCPSV}$ have a different syntax, we show that they are equally expressive. From a complexity point of view, $\mathcal{RCPSV}$ permits more compact representations than $\mathcal{RSV}$. We argue that the difference may be exponential.

1 Introduction

Although [9] early suggested the possibility of generalization to "OR" activities for the *classical* resource-constrained project scheduling ($\mathcal{RCPS}$) ([2], [6]), there were only a few publications ([4], [10]) which deal with such generalized $\mathcal{RCPS}$-problems. Further, the generalization has been mostly restricted such that only for each ground activity a set of several alternative ways of accomplishing the activity could be considered.
Newly, [5] has applied the terminological methods of KL-ONE based knowledge representation systems ([3], [7]) to project scheduling in order to formulate and solve a new general class of $\mathcal{RCPSV}$ (resource-constrained project scheduling with *variants*)-problems. $\mathcal{RCPSV}$-problems have been described as activity-terms of a terminological scheduling language called $\mathcal{RSV}$, in which the alternative processing possibilities not only for each ground activity but also for each subproject can be formulated. For each activity-term optimal active schedules

have been determined. A schedule is a set of starting times for all atomic activities in a complex activity-term, such that there are no resource conflicts, i.e. there is no moment in time with double usage of a resource. A schedule is defined to be active in the sense that no atomic activity in it can be started earlier without changing other start times.

The vocabulary of $\mathcal{RSV}$ consists of a set of ground activities $\{(i, r(i), d(i))|i = 1, \cdots, n(n \in \mathbb{N}), r(i) \in R, d(i) \in \mathbb{N}\}$ and 3 operators '**seq**' '**xor**' and '**pll**' where R is a finite set of resources. Each ground activity i is atomic and it is associated with a resource $r(i)$ and an activity time $d(i)$ needed for completing it.
The operators are used for constructing activity-terms and describing further constraints (e.g. *precedence constraint* by means of '**seq**'). Activity-terms are given inductively, just as in KL-ONE based, terminological knowledge representation languages (see e.g. $\mathcal{ALC}$[8]), as follows:

1. Each ground activity is an activity-term.
2. If $t_1, t_2, \cdots, t_k$ are activity-terms, then

$$(\mathbf{seq}\, t_1, t_2, \cdots, t_k),$$
$$(\mathbf{xor}\, t_1, t_2, \cdots, t_k),$$
$$(\mathbf{pll}\, t_1, t_2, \cdots, t_k)$$

   are activity-terms.

Instead of the set-wise interpretation being usual in knowledge representation systems, the interpretation function defined in $\mathcal{RSV}$ assigns to every term t some subset that consists of all active schedules derived from t. Based on this semantics, a calculus is defined which can transform each term t into a semantically equivalent, normalized term s. It follows from the semantical equivalence of t and s that a schedule which is optimal for t is optimal for s too and vice versa. But a normalized term is structurally simpler, i.e. in s all nonredundant *reduced* terms included in t that represent classical $\mathcal{RCPS}$-problems (scheduling problems without considering "OR" activities) and take partially different paths but complete the same project are described separately. So, for every reduced term, schedules with the minimal makespan can be computed using an algorithm for solving the classical $\mathcal{RCPS}$-problem. Among all these computed schedules, those that have the minimal value correspond then to the optimal schedules for the $\mathcal{RSV}$-term t.

In this paper we introduce a further logic-based terminological language called $\mathcal{RCPSV}$ which may be used to model a class of $\mathcal{RCPSV}$-problems. The way of construction of the language $\mathcal{RCPSV}$ and the applied methods for representing and solving the $\mathcal{RCPSV}$-problem are similar to those of $\mathcal{RSV}$.
Though $\mathcal{RSV}$ and $\mathcal{RCPSV}$ have a different syntax, we show that they are equally expressive, i.e. each $\mathcal{RCPSV}$-expression can be represented as a $\mathcal{RSV}$-expression and vice versa. But from a complexity point of view, $\mathcal{RCPSV}$ permits more compact representations than $\mathcal{RSV}$.

2 The Scheduling Language $\mathcal{RCPSV}$

Let T be a set of vertices (terms) and $E = \{(t_1, t_2), (t_3, t_4), \cdots, (t_{n-1}, t_n)\} \subset T \times T$ a set of precedence edges. Further let (T, E) be a directed acyclic graph having no edge of type (t, t) for any $t \in T$ that we call *a scheduling network on* T. If in (T, E) there exists no isolated vertex t, (T, E) can be described exclusively only through the specification of E, because T can be derived from E ($T = \{t_1, t_2\} \cup \{t_3, t_4\} \cup \cdots \cup \{t_{n-1}, t_n\}$). When in (T, E) there is an isolated vertex t, we take a dummy vertex 0 and form the pair $(0, t)$ for each isolated vertex t. Then we add $(0, t)$ to E. So we get $E'(\supset E)$ which T can be derived from. So any scheduling network (T, E) can be described exclusively only through the set of precedence edges (E or E'). This always existing simplification possibility makes a new general class of $\mathcal{RCPSV}$-problems possible to be modeled through a terminological scheduling language that we call $\mathcal{RCPSV}$.

2.1 The syntax of the language $\mathcal{RCPSV}$

Definition 1. *The vocabulary of $\mathcal{RCPSV}$ consists of two disjoint sets of symbols. These sets are:*

- *A finite set of ground activities $\{(0, eu, 0)\} \cup \{(i, r(i), d(i)) | i = 1, \cdots, n (n \in \mathbb{N}), r(i) \in R, d(i) \in \mathbb{N}\}$ where $(0, eu, 0)$ corresponds to a dummy ground activity and R is a finite set of resources. Each ground activity is atomic and is associated with a resource and an activity time needed for completing it. Except for the unlimited available dummy resource eu, each resource can be assigned to only one activity at a time (resource constraint). Activity splitting is not allowed (nonpreemptive case).*
- *A set of two structural symbols (operators) '***xor***' and '***hnet***'.*

The activity-terms of $\mathcal{RCPSV}$ are given inductively as follows :

1. *Each ground activity is an activity-term.*
2. *If $t_1, t_2, \cdots, t_k$ are activity-terms, then all terms*

$$(\mathbf{xor}\, t_1, t_2, \cdots, t_k)$$

and

$$\mathbf{hnet}[let\, n_1 = t_1, \cdots, n_k = t_k; (n_{11}, n_{12}), \cdots, (n_{j1}, n_{j2})]$$

are activity-terms where $n_1, \cdots, n_k$ are distinct constant symbols (names) and $[(n_{11}, n_{12}), \cdots, (n_{j1}, n_{j2})]$ corresponds to a scheduling network on $\{n_1, \cdots, n_k\} (= \{t_1, \cdots, t_k\})$.

The dummy ground activity $(0, eu, 0)$ corresponds to the above described dummy vertex. The operators '**xor**' and '**hnet**' are used for constructing activity-terms and have the following meaning:

– '**xor**': This operator can be used to select an activity-term among several different alternative activity-terms. *Exactly one* activity-term among alternatives must be selected and executed.
– '**hnet**': This operator specifies the arrangement of activity-terms corresponding to the given precedence relations (*precedence constraint*). In term $\mathbf{hnet}[let\, n_1 = t_1, \cdots, n_k = t_k; (n_{11}, n_{12}), \cdots, (n_{j1}, n_{j2})]$, the operator **hnet** forces $[(n_{11}, n_{12}), \cdots, (n_{j1}, n_{j2})]$ to specify a directed acyclic graph with $n_{i1} \neq n_{i2}$ for each $i = 1, \cdots k$ for the set of vertices $\{n_{11}, n_{12}\} \cup \cdots \cup \{n_{k1}, n_{k2}\}$.

2.2 The semantics of the language $\mathcal{RCPSV}$

Similar to $\mathcal{RSV}$, a $\mathcal{RCPSV}$-term s is called a *reduced* (activity-)term of a $\mathcal{RCPSV}$-term t, if s can be derived from t by replacing each term of the form $(\mathbf{xor}\, l_1, \cdots, l_n)$ in t by exactly one l_i ($i = 1, \cdots$ or n) so that s is **xor**-free. Associated with any $\mathcal{RCPSV}$-term t, there exist finitely many different reduced terms which can be derived from t. These reduced terms take partially different paths but complete the same project.
An active schedule derived from a $\mathcal{RCPSV}$-term t can be defined just as that of $\mathcal{RSV}$. Further it can be shown in the same way as in $\mathcal{RSV}$ that the set of active schedules derived from any $\mathcal{RCPSV}$-term t is finite. Now, the semantics of $\mathcal{RCPSV}$ can be defined as follows:

Definition 2. *The model-theoretic semantics of $\mathcal{RCPSV}$-activity-terms 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 $\mathcal{RCPSV}$. The interpretation function $\cdot^{\mathcal{I}}$ assigns to every activity-term t some subset of $\mathcal{D}$ that consists of all active schedules derived from t.*

2.3 The scheduling problem $\mathcal{RCPSV}$

The objective is minimizing the project makespan. So, we define the scheduling problem $\mathcal{RCPSV}$ as follows:

For a given activity-term of $\mathcal{RCPSV}$ an active schedule which has the minimal project makespan (project completion time) has to be determined.

3 A Calculus for the Scheduling Language $\mathcal{RCPSV}$

There are terms which are syntactically different, but semantically equivalent. Based on the semantics of $\mathcal{RCPSV}$, we can define a calculus called $\mathcal{RCPSV}$-calculus, which transforms a term into another semantically equivalent term of it.

Definition 3. *If $t_1, t_2, \cdots, t_k, s_1, \cdots, s_l, t_{k+2}, \cdots, t_n$ are activity-terms, the calculus has the associative rule (1) and distributive rules (2, 3) (see figure 1). The associative rule (1) describes a subexpression combined by '***xor***' which is an argument of the operator '***xor***' may be flattened. The distributive rules (2, 3) describe if a subexpression combined by '***xor***' occurs as an argument of the operator '***hnet***', the operator '***xor***' may be moved to the leftmost position.*

$$\frac{(\mathbf{xor}\, t_1, t_2, \cdots, t_k, (\mathbf{xor}\, s_1, s_2, \cdots, s_l), t_{k+2}, t_{k+3}, \cdots, t_n)}{(\mathbf{xor}\, t_1, t_2, \cdots, t_k, s_1, s_2, \cdots, s_l, t_{k+2}, t_{k+3}, \cdots, t_n)} \quad (1)$$

$$\frac{(\mathbf{hnet}[let\ n_1 = t_1, \cdots, n_{k+1} = (\mathbf{xor}\, s_1, s_2, \cdots, s_l), \cdots, n_n = t_n; (n_{11}, n_{12}), \cdots, (n_{h1}, n_{k+1}), \cdots, (n_{j1}, n_{j2})])}{\begin{array}{r}(\mathbf{xor}\, \mathbf{hnet}[let\ n_1 = t_1, \cdots, n_{k+1} = s_1, \cdots, n_n = t_n; (n_{11}, n_{12}), \cdots, (n_{h1}, n_{k+1}), \cdots, (n_{j1}, n_{j2})], \\ \vdots \\ \mathbf{hnet}[let\ n_1 = t_1, \cdots, n_{k+1} = s_l, \cdots, n_n = t_n; (n_{11}, n_{12}), \cdots, (n_{h1}, n_{k+1}), \cdots, (n_{j1}, n_{j2})])\end{array} \quad (2)$$

$$\frac{(\mathbf{hnet}[let\ n_1 = t_1, \cdots, n_{k+1} = (\mathbf{xor}\, s_1, s_2, \cdots, s_l), \cdots, n_n = t_n; (n_{11}, n_{12}), \cdots, (n_{k+1}, n_{h2}), \cdots, (n_{j1}, n_{j2})])}{\begin{array}{r}(\mathbf{xor}\, \mathbf{hnet}[let\ n_1 = t_1, \cdots, n_{k+1} = s_1, \cdots, n_n = t_n; (n_{11}, n_{12}), \cdots, (n_{k+1}, n_{h2}), \cdots, (n_{j1}, n_{j2})], \\ \vdots \\ \mathbf{hnet}[let\ n_1 = t_1, \cdots, n_{k+1} = s_l, \cdots, n_n = t_n; (n_{11}, n_{12}), \cdots, (n_{k+1}, n_{h2}), \cdots, (n_{j1}, n_{j2})])\end{array} \quad (3)$$

Figure1. Rules (1), (2), (3)

In the following we formalize the correctness of the $\mathcal{RCPSV}$-calculus and a transformation possibility for activity-terms. But we omit their proof here since they can be proved very similar to those of $\mathcal{RSV}$.

Lemma 1. *The $\mathcal{RCPSV}$-calculus is a correct calculus.*

The correctness of the $\mathcal{RCPSV}$-calculus permits to formalize the following theorem:

Theorem 1. *For any $\mathcal{RCPSV}$-term t all operators '***xor***' in the interior of t always may be moved to the leftmost position, so that t is transformed to a semantically equivalent, normalized term s in which all nonredundant reduced terms derived from t are being combined by the uniquely occurring operator '***xor***'.*

In theorem 1, it follows from semantical equivalence of a term t and its normalized term s that a schedule which is optimal for t is optimal for s too and vice versa. But for a normalized term we can consider the arguments (reduced

activity-terms) of the '**xor**'-operator separately in order to compute optimal schedules. So, because of theorem 1 the $\mathcal{RCPSV}$-problem can be solved, while first transforming each term t into a semantically equivalent, normalized term s and then computing the schedules with the minimal project makespan for every reduced term of s separately using a solution algorithm for solving the classical $\mathcal{RCPS}$-problem. The schedules among all these computed schedules that have the minimal value correspond then to the optimal schedules of t.

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

Many varieties of implicit enumeration methods ([9], [2], [6], [1]) for solving the $\mathcal{RCPS}$-problem which may be also used for determining the optimal schedules for each reduced activity-term of $\mathcal{RCPSV}$ have been reported. Further [5] has recently presented new diagram-based methods for representing and solving reduced activity-terms of $\mathcal{RSV}$. In this section we describe an algorithm for solving reduced activity-terms of $\mathcal{RCPSV}$ which is similar to her method $\mathcal{A}_{\mathcal{RSV}}$ of $\mathcal{RSV}$.

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

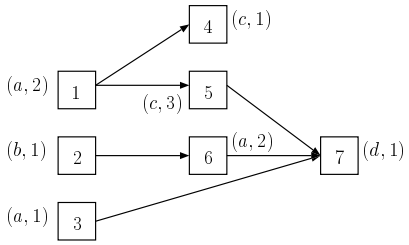

Figure2. A network

In $\mathcal{A}_{\mathcal{RCPSV}}$, another diagram than the $\mathcal{RSV}$-diagram is used as a representing method because $\mathcal{RSV}$-diagram is unsuitable to solve the $\mathcal{RCPSV}$-problem. Otherwise the same scan-line principle and the other working methods applied in $\mathcal{A}_{\mathcal{RSV}}$ are used again. For this reason, instead of a formal description for $\mathcal{A}_{\mathcal{RCPSV}}$, we only consider an example. We take the reduced activity-term represented by the network (X, P) of figure 2 with $X = \{(1, a, 2), (2, b, 1), (3, a, 1), (4, c, 1), (5, c, 3), (6, a, 2), (7, d, 1)\}$ and $P = \{(1, 4), (1, 5), (2, 6), (3, 7), (5, 7), (6, 7)\}$.

In the beginning the diagram which has a time axis and a scan-line is empty and the scan-line is found at the time $t_{SL} = 0$. In the diagram each ground activity

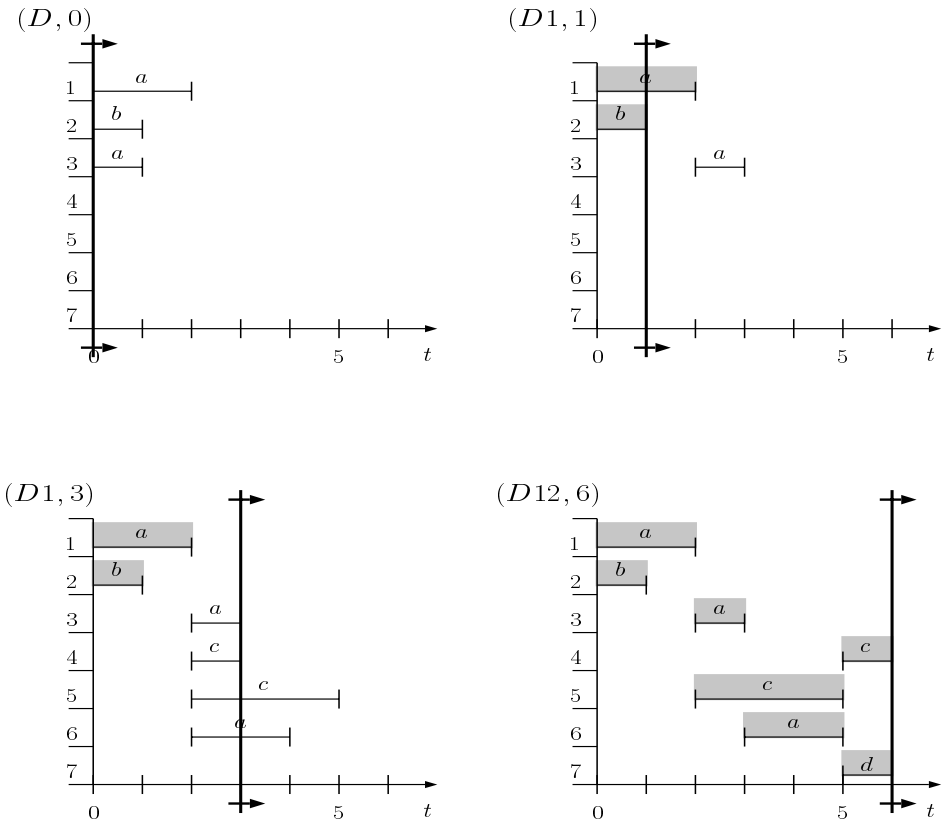

Figure3. A $\mathcal{RCPSV}$-diagram based calculation

has a left and right end point. The left and right end point of any ground activity i denoted by $LE(i)$ and $RE(i)$ 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 right stopping time while determining and then resolving resource conflicts.

Step 1: *Attaching start ground activities to the scan-line:* First all start activities out of X such as $1, 2$ and 3 which have no predecessors are attached to the scan-line. "Attaching an 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. The diagram $(D, 0)$ of figure 3 shows the resulting diagram after applying this step 1, in which the scan-line time 0 has been assigned to these start activities $1, 2$ and 3 as their 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 ground activities: There is one 1-*time conflict free activity* 2, i.e. 2 is an unique activity requiring the resource b in the time interval $[0, t_{SL}] = [0, 1]$. In addition, an 1-time resource conflict occurs since there is an activity i such that $RE(i) = t_{SL}(= 1)$ (e.g. the activity 3) and the resource $r(i)$ is required by more than one activity in the time interval $[t_{SL} - 1, t_{SL}]$. So there exist two 1-time conflict combinations [1] and [3]. The diagram $(D, 1)$ is duplicated in order to assign to each conflict combination a diagram, let these be $D1, D2$ and [1], [3] are assigned to $D1, D2$ respectively. In each diagram, the 1-time conflict free activity 2 and the assigned 1-time conflict activity are frozen in order to mark that these activities must not be moved. The other 1-time conflict activity is moved behind the frozen conflict activity. Subsequently we proceed with the next step 4 in each diagram.

If we pursue $(D1, 1)$ to which the combination [1] is assigned, we have the diagram $(D1, 1)$ of figure 3 where 1 and 2 have been frozen and 3 has been moved behind 1.

Step 4: *Attaching further ground activities to the scan-line:* Further activities out of X which can be attached to the scan-line are determined in order to place them. For an actual diagram (D, t_{SL}) and a network (X, P) a ground activity $i \in X$ can be attached to the scan-line iff 1. i isn't from the diagram (D, t_{SL}), 2. in (D, t_{SL}) there exists no *frozen* activity j with $r(i) = r(j)$ for which $LE(j) < t_{SL}$ and $RE(j) > t_{SL}$ hold and 3. all immediate predecessors e of i, i.e., for e and i $(e, i) \in P$ holds, are already elements of (D, t_{SL}) and for all e $RE(e) \leq t_{SL}$ holds in (D, t_{SL}). In $(D1, 1)$ of figure 3, the activities 4 and 5 can not be attached since it holds $RE(1) > t_{SL}(= 1)$ for their immediate predecessor 1. The activity 6 can not be attached too since in $(D1, 1)$, there exists a frozen activity 1 with $r(1) = r(6) = a$ for which $RE(1) > t_{SL}(= 1)$ holds. The resource a is being blocked until the time 2. So there is no activity to be attached to the scan-line.

Furthermore the steps 2, 3, 4 are recursively applied until all ground activities have been placed in the diagram and all activities in the diagram have been frozen so that a schedule is completed. Among all computed schedules, those that have the minimal project makespan are delivered as the optimal schedules for the network (X, P).

After applying the following steps 2, 3, 4 and 2 to the diagram $(D1, 1)$ of the last step 4, we have the diagram $(D1, 3)$ of figure 3 in which the activities $4, 5$ and 6 have been attached to the scan-line. Now, there are two 3-time conflict resources a and c. So there exist four 3-time conflict combinations $[3, 4], [3, 5], [6, 4]$ and $[6, 5]$ altogether. The diagram $(D1, 3)$ is multiplied 4 times, let these be $D11, \cdots, D14$ and $[3, 4], [3, 5], [6, 4]$, $[6, 5]$ are assigned to $D11, \cdots, D14$ respectively. If we pursue the diagram $D12$, and apply the further steps recursively, one complete active schedule is finally generated that the diagram $(D12, 6)$ of figure 3 shows. In this way, 8 nonredundant active schedules are computed altogether and there are two optimal schedules with the project makespan 6.

4.2 Proving correctness of $\mathcal{A}_{\mathcal{RCPSV}}$

We omit the proof of the following theorem, because it can be shown similar to $\mathcal{A}_{\mathcal{RSV}}$ too.

Theorem 2. *For any given reduced $\mathcal{RCPSV}$-activity-term s, $\mathcal{A}_{\mathcal{RCPSV}}$ generates all active schedules nonredundantly which may be derived from s.*

5 $\mathcal{RSV}$ and $\mathcal{RCPSV}$ are equally powerful

First we will prove formally that each $\mathcal{RCPSV}$-expression can be represented as a $\mathcal{RSV}$-expression and vice versa. It is obvious that any $\mathcal{RSV}$-expression can be represented as a $\mathcal{RCPSV}$-expression. Before showing that any $\mathcal{RCPSV}$-expression can be also represented as a $\mathcal{RSV}$-expression, we first show that any active schedule σ delivered by $\mathcal{A}_{\mathcal{RCPSV}}$ can be represented as a reduced $\mathcal{RSV}$-term $t(\sigma)$.

We consider the active schedule $\sigma = (D12, 6)$ of figure 3. With the aid of the working method of $\mathcal{A}_{\mathcal{RCPSV}}$, a reduced $\mathcal{RSV}$-term $t(\sigma) = t(\sigma_5)$ representing σ, where 5 corresponds to the number of all recursive calls of the step 3 "*Freezing*" during calculating $\sigma = (D12, 6)$, can be constructed. After each ith call of the step 3 "*Freezing*", a part σ_i of the active schedule $\sigma = (D12, 6)$ is obtained. For each $\sigma_i (i = 1, \cdots 5)$, a $\mathcal{RSV}$-activity-term $t(\sigma_i)$ representing σ_i and *consisting of all frozen activities only* can be formed so that $t(\sigma)$ representing σ finally is obtained.
After the 1st call of the step "*Freezing*", we get the diagram $\sigma_1 = (D1, 1)$ of figure 3. The $\mathcal{RSV}$-expression $t(\sigma_1)$ consists of the both frozen activities 1 and 2 (Only the frozen activities are considered.). Since it is still unknown whether the activities 1 and 2 will take successors, they are combined by '**seq**' and so we obtain **seq** 1 and **seq** 2. These activities are carried out parallel each other. So, they are combined by '**pll**' again and the following $\mathcal{RSV}$-expression finally can be obtained as $t(\sigma_1)$:

$$\begin{array}{ll} (\mathbf{pll} & (\mathbf{seq}\,1), \\ & (\mathbf{seq}\,2)) \end{array} \tag{4}$$

For $t(\sigma_2)$, it obviously holds $t(\sigma_2) = t(\sigma_1)$. At the 3rd call, the both activities 3 and 5 are frozen for which it holds $LE(3) = LE(5) = 2$. For each activity k frozen at the ith call, it holds that *either* k is a start-activity such as the activities 1 and 2, i.e. in σ_i it holds $LE(k) = 0$ *or* in σ_i there exists at least one frozen immediate predecessor v of k with $RE(v) = LE(k)$. For the both activities 3 and 5 frozen at the 3rd call, there exists such a frozen activity 1 with $RE(1) = LE(3) = LE(5)$. First, the following subactivity is constructed:

$$\begin{array}{ll} (\mathbf{pll} & (\mathbf{seq}\,3), \\ & (\mathbf{seq}\,5)) \end{array}$$

Then this subactivity is added to (4) behind the activity 1. So, for $t(\sigma_3)$ we get

$$(\mathbf{pll}\ (\mathbf{seq}\,1,\quad (\mathbf{pll}\ (\mathbf{seq}\,3),\ (\mathbf{seq}\,5))),\ (\mathbf{seq}\,2))$$

If, at the 3rd call, a further activity k with $LE(k) = 0$ had been frozen, k was combined by '**seq**' and then $\mathbf{seq}\,k$ was added to $t(\sigma_2)$ as an argument of the operator '**pll**' of (4). At the 4th call, the activity 6 is added and after the last (5th) call we get the schedule $\sigma = (D12, 6)$ and the following $\mathcal{RSV}$-expression $t(\sigma)$ representing σ can be constructed:

$$(\mathbf{pll}\ (\mathbf{seq}\,1,\quad (\mathbf{pll}\ (\mathbf{seq}\,3, 6),\ (\mathbf{seq}\,5, (\mathbf{pll}\,4, 7)))),\ (\mathbf{seq}\,2))$$

The activities $6, 4, 7$ have not been combined by '**seq**' because they take no successor.

Lemma 2. *For a network (X, P) of a given $\mathcal{RCPSV}$-expression, let σ be any complete active schedule delivered through $\mathcal{A}_{\mathcal{RCPSV}}$ and n be the number of recursive calls of the step 3 "*Freezing all definitely placed ground activities*" of $\mathcal{A}_{\mathcal{RCPSV}}$ during calculating σ. Further let $\sigma_1, \cdots, \sigma_n = \sigma$ be the sequence of partial schedules delievered after every call of the step 3 "*Freezing*", where, for every σ_i, only all frozen activities until the i-th call are considered. Then a sequence $t(\sigma_0), t(\sigma_1), \cdots t(\sigma_n)$ of $\mathcal{RSV}$-expressions representing $\sigma_1, \cdots, \sigma_n$ respectively can be constructed and so $t(\sigma_n)$ exactly represents the schedule σ.*

Proof. It can be shown easily by induction on the term $t(\sigma_i)$ of the sequence $t(\sigma_0), t(\sigma_1), \cdots t(\sigma_n)$. At the extension of the activity $t(\sigma_i)$ to the activity $t(\sigma_{i+1})$ (see the example described above), the semantics of the operators '**pll**' and '**seq**' make it possible to force the activities $d_1, d_2, \cdots, d_m$ frozen at the $(i+1)$th call to have exactly the same time intervals for carrying out just as in σ_{i+1}. □

Theorem 3. *For any $\mathcal{RCPSV}$-activity-term s there exist a $\mathcal{RSV}$-activity-term t with $s^{\mathcal{I}} = t^{\mathcal{I}}$.*

Proof. For any $\mathcal{RCPSV}$-activity-term s all nonredundant active schedules can be derived by means of $\mathcal{A}_{\mathcal{RCPSV}}$. In Lemma 2 we showed any active schedule derived from s can be represented as a $\mathcal{RSV}$-expression. Let $\tau_1, \cdots \tau_n$ be all nonredundant active schedules derived from s and $t(\tau_1), \cdots, t(\tau_n)$ be all $\mathcal{RSV}$-expressions representing these schedules $\tau_1, \cdots \tau_n$ respectively. For the $\mathcal{RSV}$-expression

$$\mathbf{xor}\, t(\tau_1), \cdots, t(\tau_n)$$

it obviously holds $s^{\mathcal{I}} = (\mathbf{xor}\, t(\tau_1), \cdots, t(\tau_n))^{\mathcal{I}}$. □

It may be noticed that in theorem 3 the expression $(\mathbf{xor}\, t(\tau_1), \cdots, t(\tau_n))$ presumably can have an exponentially larger space demand than the activity-term s. So, if we compare the languages $\mathcal{RSV}$ and $\mathcal{RCPSV}$ from the viewpoint of a syntactical representation ability of problems, we can show that $\mathcal{RCPSV}$ permits more compact representations than $\mathcal{RSV}$.

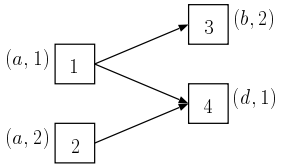

Figure4. A $\mathcal{RCPS}$-problem

We consider the $\mathcal{RCPS}$-problem which figure 4 shows. This problem can be represented by the $\mathcal{RCPSV}$-activity-term (5) in a *compact* form which we can construct simply and directly formulating precedence edges, e. g. $((1, a, 1), (3, b, 2))$, and then combining them with the operator '**hnet**'.

$$(\mathbf{hnet}[((1, a, 1), (3, b, 2)), ((1, a, 1), (4, d, 1)), \\ ((2, a, 2), (4, d, 1))]) \tag{5}$$

In order to represent this problem as a $\mathcal{RSV}$-activity-term, first, for the problem all nonredundant active schedules should be computed and then, with the aid of the construction rule described above, a corresponding $\mathcal{RSV}$-expression such as (6) can be described where each computed schedule is represented as a $\mathcal{RSV}$-reduced activity-term and all these $\mathcal{RSV}$-reduced activity-terms are then combined by '**xor**'.

$$(\mathbf{xor}\ (\mathbf{seq}\,(1, a, 1), (\mathbf{pll}\,(3, b, 2), (\mathbf{seq}(2, a, 2), (4, d, 1))), \\ (\mathbf{seq}\,(2, a, 2), (1, a, 1), (\mathbf{pll}\,(3, b, 2), (4, d, 1)))) \tag{6}$$

It can be examined easily that there is no more compact $\mathcal{RSV}$-expression representing the $\mathcal{RCPS}$-problem of figure 4 than the expression (6).

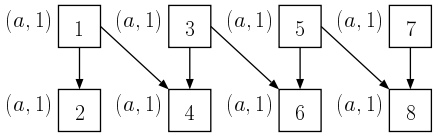

Figure5. A $\mathcal{RCPS}$-problem

Obviously, any $\mathcal{RCPS}$-problem can be translated into a $\mathcal{RCPSV}$- term in polynomial size. But we presume that there are $\mathcal{RCPS}$-problems which can be translated into $\mathcal{RSV}$-terms only in an exponential size. For example, figure 5 presumably shows such a problem.

6 Summary and Future Work

Another terminological scheduling language $\mathcal{RCPSV}$, similar to $\mathcal{RSV}$ ([5]), has been presented which may be used to formulate and solve a new general class of $\mathcal{RCPSV}$-problems. The terminological logic $\mathcal{RCPSV}$ has offered an effective approach for solving the $\mathcal{RCPSV}$-problem. A solution algorithm $\mathcal{A}_{\mathcal{RCPSV}}$ based on a scan-line principle has been introduced, through which all active schedules could be generated nonredundantly for any reduced activity term. It has been formally shown that each $\mathcal{RCPSV}$-expression can be represented as a $\mathcal{RSV}$-expression and vice versa but, from a complexity point of view, $\mathcal{RCPSV}$ permits more compact representations than $\mathcal{RSV}$.

For the future work, we can concentrate on a generalization of resource availability such that multiple units of resources and multiple number of resource types may be required by a ground activity. Such general problems may be formulated easily while only the syntax $(i, r(i), d(i))$ of constructing each ground activity is generalized to $(i, (r_1(i), \cdots, r_n(i)), d(i))$, where n corresponds to the number of resource types and $r_k(i)(k = 1, \cdots, n)$ with $0 \leq r_k(i) \leq b_k$ describes required units of resource type k by i. Here, each resource type $k(k = 1, \cdots, n)$ is assumed to be available in a constant amount b_k throughout the duration of the project. Otherwise the two structural symbols (operators) '**hnet**' and '**xor**' and the inductive rules for constructing activity-terms may be applied unchanged. Therefore, the subject of future work might be to adapt the algorithm $\mathcal{A}_{\mathcal{RCPSV}}$ to this more general problem formulation and to optimize the algorithm using additional bounding or heuristic rules so that it may be used practically for large projects.

References

1. 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.
2. E. Demeulemeester and W. Herroelen. New Benchmark Results for the Resource-constrained Project Scheduling Problem. *Management Science*, 43(11):1485–1492, 1997.
3. F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The Complexity of Concept Languages. *Information and Computation*, 134(1):1–58, 1997.
4. S. E. Elmaghraby. *Activity Networks: Project Planning and Control by Network Models*. Wiley, New York, 1977.

5. P. S. Kim. *Terminologische Sprachen zur Repräsentation und Lösung von ressourcenbeschränkten Ablaufplanungsproblemen mit Prozeßvarianten*. PhD thesis, Institut für Wirtschaftsinformatik, Universität Frankfurt, 2001.
6. 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.
7. B. Nebel and K. von Luck. Hybrid Reasoning in BACK. In Z. W. Ras and L. Saitta, editors, *Methodologies for Intelligent Systems*, pages 260–269. North Holland, Amsterdam, Netherlands, 1988.
8. 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.
9. L. Schrage. Solving Resource-Constrained Network Problems by Implicit Enumeration-Nonpreemptive Case. *Operations Research*, 10:263–278, 1970.
10. F. B. Talbot. Resource-Constrained Project Scheduling with Time-Resource Tradeoffs: The Nonpreemptive Case. *Management Science*, 28(10):1197–1210, 1982.