A Term-Based Approach to Project Scheduling

Pok-Son Kim, Manfred Schmidt-Schauß

Fachbereich Informatik
Johann Wolfgang Goethe-Universität
Postfach 11 19 32
D-60054 Frankfurt/Main, Germany
Tel.: (+49) 69 798 28597; Fax: (+49) 69 798 28919
`{kim, schauss}@cs.uni-frankfurt.de`

Abstract We introduce a new method for representing and solving a general class of non-preemptive resource-constrained project scheduling problems. The new approach is to represent scheduling problems as descriptions (activity terms) in a language called $\mathcal{RSV}$, which allows nested expressions using **pll**, **seq**, and **xor**. The activity-terms of $\mathcal{RSV}$ are similar to concepts in a description logic. The language $\mathcal{RSV}$ generalizes previous approaches to scheduling with variants insofar as it permits **xor**'s not only of atomic activities but also of arbitrary activity terms. A specific semantics that assigns their set of active schedules to activity terms shows correctness of a calculus normalizing activity terms $\mathcal{RSV}$ similar to propositional DNF-computation.
Based on $\mathcal{RSV}$, this paper describes a diagram-based algorithm for the $\mathcal{RSV}$-problem which uses a scan-line principle. The scan-line principle is used for determining and resolving the occurring resource conflicts and leads to a nonredundant generation of all active schedules and thus to a computation of the optimal schedule.

1 Introduction

Ever since the introduction of the pioneer works of Kelly [13] and Wiest [27] very much has been reported for the *classical* resource-constrained project scheduling ($\mathcal{RCPS}$) but most approaches solving this problem use either heuristic algorithms ([11], [5], [3], [14], [19]) or exact algorithms ([21], [23], [24], [26], [7], [8], [18]), are restricted to the case in which each job could be performed in only one prescribed way. Although Schrage [23] and König et al. [16] suggested the possibility of generalization to "OR" activities, there are only a few publications ([10], [25]) which deal with a class of resource-constrained scheduling problems with *variants*, where "OR" was restricted to atomic activities.
This paper generalizes *r*esource constrained project *s*cheduling with *v*ariants ($\mathcal{RSV}$) to allow "OR" of compound activities. The use of semantic methods from description logics ([22,9] is the key for understanding the meaning of compound activity terms (for more information see [15]).
Description Logic emerged from KL-ONE-based, terminological knowledge representation systems and is now an active field of research in artificial intelligence

with its own conference(s). Characteristics of descriptions logics are a term language for concepts and other notions, a clean denotational semantics, and specific calculi (like subsumption) based on the semantics. There are extension of description logics by temporal operators (see e.g. [1,2]), which allow reasoning about concepts in time.

Scheduling is roughly the problem, given certain tasks, their duration and resource usage to find an optimal conflict free schedule, usually on a discrete time line. It is possible to encode these problems into temporal description logics, though it appears to be inconvenient, since for scheduling problems, time has to be measured and added, and the goal is finding an optimal schedule. Furthermore reasoning is not optimized to deal with the constraint of exclusive resource usage.

The methods proposed so far for solving scheduling problems are mostly based on integer programming. In that approach it is generally difficult to read the flow structure and the content of a scheduling problem (for example, which activity requires what resource).

This motivates to use a term language $\mathcal{RSV}$ for activity terms and to model its semantics in a way best suited to the specific time and resource constraints. Based on this semantics, a calculus is defined which can transform each activity term A into a semantically equivalent, normalized activity term B, which is a nonredundant disjunction of *reduced* activity terms corresponding to A. For every reduced activity term, optimal schedules (with the minimal makespan) can be computed using an algorithm for solving the classical $\mathcal{RCPS}$-problem. For any $\mathcal{RSV}$-activity term the optimal schedules can be computed by this two-step process using a final minimization. It is non-obvious how to compute the active schedules of compound activity terms in another way.

Further we introduce a new diagram-based method, which represents reduced activity terms graphically as $\mathcal{RSV}$-diagrams. Based on this method, an algorithm for generating all *nonredundant* active schedules for a reduced activity term is described, which uses a time-based scan-line principle to determine and resolve the occurring resource conflicts. The principle of using a scan-line is well known in the area of geometrical algorithms ([20], [28]) and may be used for example to solve problems occurring during the design of VLSI circuits.

This paper is structured as follows. First, the scheduling language $\mathcal{RSV}$ is defined in section 2. Then the calculus is described in section 3. The optimal solution algorithm and the correctness proof for it are given in section 4. The last section 5 contains conclusions and future work.

2 The Scheduling Language $\mathcal{RSV}$

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

2.1 The Syntax of the Language $\mathcal{RSV}$

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

– *A set of* atomic activities*, also called* ground activities*. Each atomic activity consists of a name and two integer constants (written like arguments of a predicate symbol). The first denotes a resource; the second a duration. For P a name and $r, t \in \mathbb{N}$, $P(r, t)$ is a ground activity. The name of each ground activity is uniquely chosen. For a ground activity A let $r(A)$ and $d(A)$ denote its associated resource and duration, respectively.*
– *Three structural symbols (operators)* **seq**, **xor** *and* **pll**.

The activity terms of $\mathcal{RSV}$ are inductively defined as follows:

1. *Each ground activity is an activity term.*
2. *If $A_1, A_2, \cdots, A_k$ are activity terms, then*
$$(\mathbf{seq}\, A_1, A_2, \cdots, A_k),$$
$$(\mathbf{xor}\, A_1, A_2, \cdots, A_k),$$
$$(\mathbf{pll}\, A_1, A_2, \cdots, A_k)$$
are also activity terms.

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

– '**seq**' : This operator specifies the *sequential* processing of an activity term or activity terms (precedence constraints).
– '**xor**' : This operator can be used to select an activity term among several different alternative activity terms. *Exactly one* activity term among the alternatives must be selected and executed.
– '**pll**' : This operator specifies the possibility of parallel processing of activity terms.

2.2 Reduced Activity Terms

An expression of $\mathcal{RSV}$ which is **xor**-free is called *a reduced activity term.* For a project represented by a reduced activity term there is no selection possibility for any part of it.
B is a *reduced activity subterm of* A, if B can be derived from A by repeatedly replacing subterms of the form $(\mathbf{xor}\, C_1, \cdots, C_n)$ by exactly one C_i ($i = 1, \cdots$ or n) so that B is **xor**-free. Associated with any activity term A of $\mathcal{RSV}$, there exist finitely many different reduced activity terms which can be derived from A. These reduced activity terms take partially different paths but complete the same project.

Example 1. We show how to encode the small project of preparing food for two persons and then eating this food. The project is to either putting 2 pizzas in the oven, or putting 4 toasts into the toaster, and then eating them. Let resource

1 be the oven, resource 2 be the toaster, 3 and 4 are the two persons. Then a description could be

$$\begin{array}{l}(\textbf{seq}\ (\textbf{xor}\ \texttt{pizza}(1,15)\ (\textbf{pll}\ \texttt{toastfst}(2,2),\ \texttt{toastsnd}(2,2))))\\ \quad(\textbf{pll}\ \texttt{arne\_eating}(3,10),\ \texttt{pokson\_eating}(4,12))\end{array}$$

Note that we permit several occurrences of the same ground activity in an activity term but this is not permitted for a reduced activity term, i.e. each ground activity can occur once and only once in a reduced activity term.

2.3 Schedules

For a reduced activity term A let $g(A) = \{A_1, \cdots, A_n\}$ be the set consisting of all ground activities occurring in A. The activity term A defines a strict partial order $<_A$ on $\{A_1, \cdots, A_n\}$, using the sequentiality operator. It is generated on the set S of subterms of A as follows:

- $(\textbf{seq}\ B_1, \ldots, B_m) \in S \wedge i < j \Rightarrow B_i <_A B_j$.
- $B_1, B_2 \in S \wedge B_1 <_A B_2 \Rightarrow B'_1 <_A B'_2$ for every subterm B'_i of $B_i, i = 1, 2$.

Definition 2. *Let A be a reduced activity term and $g(A) = \{A_1, \cdots, A_n\}$.*
An active schedule *for A is a set of starting times of ground activities $\{t_{A_i} \in \mathbb{N} \mid A_i \in g(A)\}$ such that:*

- *The precedence constraints are satisfied: $t_{A_h} + d(A_h) \leq t_{A_i}$ for each A_i and each immediate predecessor A_h with $A_h <_A A_i$,*
- *The resource constraints are satisfied: $t_{A_m} \geq t_{A_l} + d(A_l)$ or $t_{A_l} \geq t_{A_m} + d(A_m)$ for all $A_l, A_m \in g(A)$ with $r(A_l) = r(A_m) (l \neq m)$ and*
- *No ground activity can be started earlier without changing other start times: There does not exist another set $\{t'_{A_i} | A_i \in g(A)\}$ with a ground activity A_j, which satisfies the precedence and resource constraints, such that $t_{A_i} = t'_{A_i}$ for $i \neq j$ and $t_{A_j} > t'_{A_j}$.*

The makespan *of an active schedule is the duration from the first starting time $\texttt{min}_i(t_{A_i})$ to the stopping time $\texttt{max}_i(t_{A_i} + d(A_i))$.*
Let A be an activity term. Then the set of active schedules for A is the union of the set of active schedules for all reduced activity subterms of A.

Since time is discrete it is easy to see that for any activity term A the set of active schedules derived from A is finite. In the following all schedules are assumed to be active.

2.4 The Semantics of the Language $\mathcal{RSV}$

Definition 3. *The model-theoretic semantics of activity terms in $\mathcal{RSV}$ 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 $\mathcal{RSV}$. The interpretation function $\cdot^{\mathcal{I}}$ assigns to every activity term A the subset of $\mathcal{D}$ that consists of all active schedules derived from A.*

2.5 Scheduling Problem

The objective is minimizing the project makespan, i.e. finding an active schedule with a minimal duration. So, we define the scheduling problem $\mathcal{RSV}$ as follows:

Definition 4 (The scheduling problem $\mathcal{RSV}$).
For a given activity term A of $\mathcal{RSV}$ an (active) schedule corresponding to A which has the minimal project makespan has to be determined.

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

Two activity terms are semantically equivalent, if the interpretations of the two activity terms are identical. For example, the following two activity terms

$$(\mathbf{pll}\,(\mathbf{seq}\,P_3(c,15),P_4(c,16)),(\mathbf{xor}\,P_5(c,3),P_6(d,5))) \tag{1}$$

and

$$\begin{aligned}(\mathbf{xor}\ &(\mathbf{pll}\,(\mathbf{seq}\,P_3(c,15),P_4(c,16)),P_5(c,3)),\\ &(\mathbf{pll}\,(\mathbf{seq}\,P_3(c,15),P_4(c,16)),P_6(d,5)))\end{aligned} \tag{2}$$

are semantically equivalent, i.e. the set of all schedules which may be derived from (1) and the set of all schedules which may be derived from (2) are identical. An activity term such as (2) in which the **xor**-operator occurs only once in the leftmost position is called a *normalized* activity term. In this section we will define a calculus that may be used to transform any $\mathcal{RSV}$-expression A into a semantically equivalent *normalized* $\mathcal{RSV}$-expression B. Further we will show the calculus to be correct, i.e. in the calculus only such syntactical derivations which cause no semantical change are permitted. So, if $A \vdash B$, then $A \doteq B$ holds, as illustrated in Figure 1.

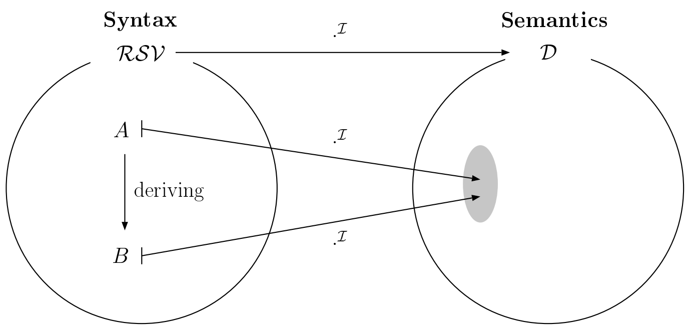

Figure1. Graphical representation of a scheduling equation $A \doteq B$

3.1 The $\mathcal{RSV}$ -calculus

Similar to the computation rules for a disjunctive normal form in propositional logic, a $\mathcal{RSV}$- calculus is defined:

Definition 5. *If $A_1, A_2, \cdots, A_k, B_1, \cdots, B_l, A_{k+2}, \cdots, A_n$ are activity terms, the calculus has the following 3 associative rules (3,4,5) and 2 distributive rules (6,7). Each associative rule describes that a subexpression combined by '**seq**', '**xor**' or '**pll**' which is an argument of the operator '**seq**', '**xor**' or '**pll**' respectively may be flattened. Each distributive rule describes that if the operator '**xor**' occurs as an argument of the operator '**seq**' or '**pll**', the operator '**xor**' may be moved to the leftmost position:*

$$\frac{(\mathbf{seq}\, A_1, A_2, \cdots, A_k, (\mathbf{seq}\, B_1, B_2, \cdots, B_l), A_{k+2}, A_{k+3}, \cdots, A_n)}{(\mathbf{seq}\, A_1, A_2, \cdots, A_k, B_1, B_2, \cdots, B_l, A_{k+2}, A_{k+3}, \cdots, A_n)} \tag{3}$$

$$\frac{(\mathbf{xor}\, A_1, A_2, \cdots, A_k, (\mathbf{xor}\, B_1, B_2, \cdots, B_l), A_{k+2}, A_{k+3}, \cdots, A_n)}{(\mathbf{xor}\, A_1, A_2, \cdots, A_k, B_1, B_2, \cdots, B_l, A_{k+2}, A_{k+3}, \cdots, A_n)} \tag{4}$$

$$\frac{(\mathbf{pll}\, A_1, A_2, \cdots, A_k, (\mathbf{pll}\, B_1, B_2, \cdots, B_l), A_{k+2}, A_{k+3}, \cdots, A_n)}{(\mathbf{pll}\, A_1, A_2, \cdots, A_k, B_1, B_2, \cdots, B_l, A_{k+2}, A_{k+3}, \cdots, A_n)} \tag{5}$$

$$\frac{(\mathbf{seq}\, A_1, A_2, \cdots, A_k, (\mathbf{xor}\, B_1, B_2, \cdots, B_l), A_{k+2}, A_{k+3}, \cdots, A_n)}{\begin{array}{r}(\mathbf{xor}\ (\mathbf{seq}\, A_1, A_2, \cdots, A_k, B_1, A_{k+2}, A_{k+3}, \cdots, A_n),\\ (\mathbf{seq}\, A_1, A_2, \cdots, A_k, B_2, A_{k+2}, A_{k+3}, \cdots, A_n),\\ \vdots \qquad\qquad\quad\\ (\mathbf{seq}\, A_1, A_2, \cdots, A_k, B_l, A_{k+2}, A_{k+3}, \cdots, A_n))\end{array}} \tag{6}$$

$$\frac{(\mathbf{pll}\, A_1, A_2, \cdots, A_k, (\mathbf{xor}\, B_1, B_2, \cdots, B_l), A_{k+2}, A_{k+3}, \cdots, A_n)}{\begin{array}{r}\mathbf{xor}\ ((\mathbf{pll}\, A_1, A_2, \cdots, A_k, B_1, A_{k+2}, A_{k+3}, \cdots, A_n),\\ (\mathbf{pll}\, A_1, A_2, \cdots, A_k, B_2, A_{k+2}, A_{k+3}, \cdots, A_n),\\ \vdots \qquad\qquad\quad\\ (\mathbf{pll}\, A_1, A_2, \cdots, A_k, B_l, A_{k+2}, A_{k+3}, \cdots, A_n))\end{array}} \tag{7}$$

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

Proof. A rule in the form

$$\frac{A}{B}$$

is "correct" iff the interpretation of the upper expression A and the lower expression B is identical ($A^{\mathcal{I}} = B^{\mathcal{I}}$). In the following we show this for each of the 5 rules.

Rule (3): The set of all active schedules derived from the upper and lower expression is obviously identical, because both expressions describe the same ordering of the activity terms $A_1, \cdots, A_k, B_1, \cdots, B_l, A_{k+2}, \cdots, A_{n-1}$ and A_n. Otherwise the expression transformation doesn't make changes. Therefore, the following equation holds:

$$\begin{aligned}&(\mathbf{seq}\, A_1, A_2, \cdots, A_k, (\mathbf{seq}\, B_1, B_2, \cdots, B_l), A_{k+2}, A_{k+3}, \cdots, A_n)^{\mathcal{I}}\\ &\quad = (\mathbf{seq}\, A_1, A_2, \cdots, A_k, B_1, B_2, \cdots, B_l, A_{k+2}, A_{k+3}, \cdots, A_n)^{\mathcal{I}}\end{aligned}$$

Rule (4) and *Rule* (5) may be proved similarly to *rule* (3).
Rule (6): For the $k+1$-st argument of the operator '**seq**' a choice possibility exists. One of the l activity terms $B_1, \cdots B_{l-1}$ and B_l must be selected. For each choice of B_i $(i = 1, \cdots, l)$ a set of all schedules derived from the expression

$$(\mathbf{seq}\, A_1, A_2, \cdots, A_k, B_i, A_{k+2}, A_{k+3}, \cdots, A_n)$$

denoted by $M_{B_i}$is determined. Then the set of all schedules which may be derived from the upper expression

$$(\mathbf{seq}\, A_1, A_2, \cdots, A_k, (\mathbf{xor}\, B_1, B_2, \cdots, B_l), A_{k+2}, A_{k+3}, \cdots, A_n)$$

corresponds to the union of the sets $M_{B_1} \cdots M_{B_{l-1}}$ and M_{B_l}. Further this union corresponds to the set of all schedules which may be derived from the lower expression. So, the following equation holds:

$$\begin{aligned}&(\mathbf{seq}\, A_1, A_2, \cdots, A_k, (\mathbf{xor}\, B_1, B_2, \cdots, B_l), A_{k+2}, A_{k+3}, \cdots, A_n)^{\mathcal{I}}\\ &\quad = (\mathbf{xor}\ (\mathbf{seq}\, A_1, A_2, \cdots, A_k, B_1, A_{k+2}, A_{k+3}, \cdots, A_n),\\ &\qquad\qquad (\mathbf{seq}\, A_1, A_2, \cdots, A_k, B_2, A_{k+2}, A_{k+3}, \cdots, A_n),\\ &\qquad\qquad\qquad \vdots\\ &\qquad\qquad (\mathbf{seq}\, A_1, A_2, \cdots, A_k, B_l, A_{k+2}, A_{k+3}, \cdots, A_n))^{\mathcal{I}}\end{aligned}$$

Rule (7): This may be proved similar to *rule* (6).

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

Theorem 1. *For any activity description A of $\mathcal{RSV}$ all operators '***xor***' in the interior of A always can be moved to the leftmost position such that A is transformed to a semantically equivalent, normalized activity term A' in which the operator '***xor***' can occur uniquely once in the leftmost position combining all reduced activity terms derived from A.*

Proof. It is sufficient to show that for any expression A of $\mathcal{RSV}$ all derivations terminate in a normalized expression. This is the case when no further $\mathcal{RSV}$-rules can be applied.
Using an innermost strategy and induction on the number of occurrences of **xor**'s it is easy to see that in every expression containing the '**xor**'-operator, it can be shifted to the topmost position.
Thus all '**xor**'-operators in the interior of the activity term A may be moved stepwise to the left until at most one topmost **xor** remains. □

Theorem 1 shows that the $\mathcal{RSV}$-problem can be solved by first transforming each activity term A into a semantically equivalent, normalized activity term B and then computing the schedules with the minimal project makespan for every reduced activity term of B separately. A final minimizing step computes the minimum makespan for A.
In the following section we describe a solution algorithm that nonredundantly computes all active schedules for each reduced activity term of $\mathcal{RSV}$.

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

Many varieties of branch-and-bound-based implicit enumeration methods ([23], [24], [26], [4], [7], [8], [18], [6]) for solving the $\mathcal{RCPS}$-problem which may be also used for determining the optimal schedules for reduced activity terms of $\mathcal{RSV}$ have been reported. In this section we introduce a new diagram-based method for representing reduced activity terms graphically. The resulting diagrams are called $\mathcal{RSV}$-diagrams. Further we show that based on the representation method, a solution algorithm is described for explicit generation of all nonredundant active schedules. This is illustrated graphically using $\mathcal{RSV}$-diagrams.

4.1 The $\mathcal{RSV}$-Diagram

A $\mathcal{RSV}$-diagram has a time axis and a scan-line. The operator '**seq**' is specified using a continuous line while the operator '**pll**' is specified using a broken line. In the following two reduced activity terms are for example represented by a $\mathcal{RSV}$-diagram.

Example 2. The following two reduced activity terms

$$\begin{array}{l}\mathbf{pll}\,(\mathbf{seq}\,P_1(b,3),P_2(c,4)),\\ \quad(\mathbf{seq}\,P_3(a,2),P_4(b,3)),\\ \quad(\mathbf{seq}\,P_5(a,4),P_6(b,2))\end{array} \quad\text{and}\quad \begin{array}{l}\mathbf{pll}\,(\mathbf{seq}\,(\mathbf{pll}\,P_1(a,1),P_2(b,2)),P_3(c,2)),\\ \quad(\mathbf{seq}\,P_4(b,1),P_5(a,1)),\\ \quad P_6(a,2),\\ \quad(\mathbf{seq}\,P_7(d,1),P_8(b,2))\end{array}$$

may be represented by the $\mathcal{RSV}$-diagram 1 and 2 of Figure 2 respectively.

4.2 Solution Algorithm $\mathcal{A}_{\mathcal{RSV}}$ Based on a Scan-Line principle

In a $\mathcal{RSV}$-diagram each ground activity has a left and a right end point (a start and end time). The left and right end point of any ground activity $P(r,t)$ denoted by $LE(P(r,t))$ and $RE(P(r,t))$ are referred to as *stopping times* of the scan-line. (D,t) with $t \geq 0$ denotes that the scan-line is found at the stopping time $t_{SL} = t$ in the $\mathcal{RSV}$-diagram D. The scan-line is used for determining and resolving resource conflicts. Instead of continuously moving, the scan-line jumps from one stopping time to the next right stopping time while determining and then resolving resource conflicts.

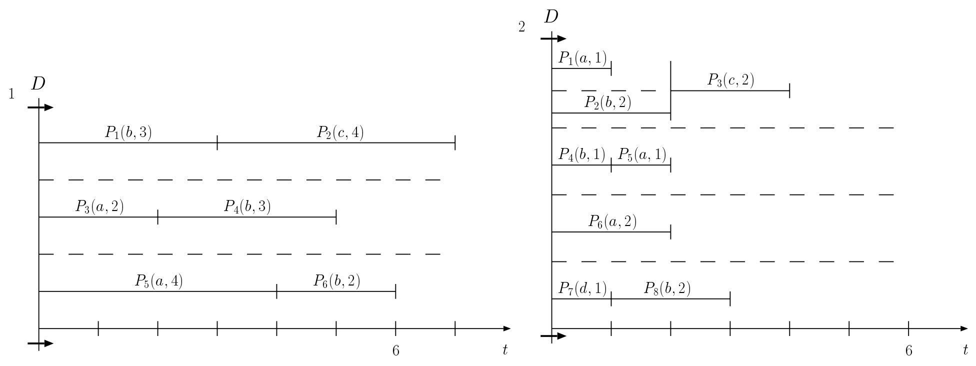

Figure2. $\mathcal{RSV}$-diagrams with the scan-line on the stopping time $t_{SL} = 0$.

In the beginning the scan-line is found at the time $t_{SL} = 0$ and the diagram is empty. The diagram is accompanied by a given input activity term A and an actual activity term T, where in the beginning $A = T$ holds.
Step 1: **Attaching start ground activities to the scan-line:** First all start ground activities of A (or T) which have no predecessors in A are attached to the scan-line. "Attaching a ground activity P to the scan-line" means that P is placed in the diagram so that the time at which the scan-line is found is assigned to P as its start time.
Step 2: **Moving the scan-line:** The scan-line jumps to the next stopping time.

Definition 6. *In a $\mathcal{RSV}$-diagram (D, t_{SL}) with $t_{SL} > 0$ a ground activity $P(r,t)$ is called a (t_{SL}-time) scan-line activity iff $LE(P(r,t)) < t_{SL}$ and $RE(P(r,t)) \geq t_{SL}$ holds. A scan-line activity $P(r,t)$ with $RE(P(r,t)) = t_{SL}$ is called a (t_{SL}-time) direct scan-line activity. A direct scan-line activity $P(r,t)$ is called a (t_{SL}-time) conflict-free activity iff there is no other scan-line activity that requires the resource r, i.e. $P(r,t)$ is a unique scan-line activity requiring the resource r. The resource r which is required by a direct scan-line activity is called a (t_{SL}-time) direct scan-line resource.*

Definition 7. *If in a $\mathcal{RSV}$-diagram (D, t_{SL}) with $t_{SL} > 0$, ground activities $P_1(r,t_1), P_2(r,t_2), \cdots$ and $P_n(r,t_n) (n \geq 2)$ and are all (t_{SL}-time) scan-line activities requiring the same resource r, $P_1(r,t_1), P_2(r,t_2), \cdots$ and $P_n(r,t_n)$ $(n \geq 2)$ are called to be involved in a (t_{SL}, r)-resource conflict or t_{SL}-time resource conflict iff r is a direct scan-line resource . This resource r is called a t_{SL}-time conflict resource. Further tthese activities $P_1(r,t_1), P_2(r,t_2), \cdots$ and $P_n(r,t_n)$ $(n \geq 2)$ are called (t_{SL}, r)-conflict activities or t_{SL}-time conflict activities.*

Step 3: **Determining and resolving resource conflicts; Freezing all definitely placed ground activities:** First, because the begin and end times of all t_{SL}-time conflict-free activities have been definitely determined, all t_{SL}-time conflict-free activities are frozen. If several scan-line activities require a conflict

resource r simultaneously, a resource conflict occurs. A resource conflict is resolved by selecting an activity and shifting all the other activities behind the selected activity. In this case the begin and end time of this selected activity are definitely determined. In order to mark that a selected activity must no longer be moved, 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 it. There exist several different combinational possibilities for selecting activities. Such a combination is called *a conflict combination* and is formally defined as follows:

Definition 8. *Let $r_1, r_2, \cdots, r_n$ be t_{SL}-time conflict resources in a $\mathcal{RSV}$-diagram (D, t_{SL}). Let $P_{r_1,1}, \cdots, P_{r_1,m_1}$ be all (t_{SL}, r_1)-conflict activities, $P_{r_2,1}, \cdots, P_{r_2,m_2}$ be all (t_{SL}, r_2)-conflict activities, $\cdots$, and $P_{r_n,1}, \cdots, P_{r_n,m_n}$ be all (t_{SL}, r_n)-conflict activities. An element of the following set*

$$\{[P_{r_1,i_1}, P_{r_2,i_2}, \cdots, P_{r_n,i_n}] | i_1 = 1, \cdots, m_1, i_2 = 1, \cdots, m_2, \cdots, i_n = 1, \cdots, m_n\}$$

is called a t_{SL}-time conflict combination.

In the case of the definition 8 there exist altogether $m_1 \times m_2 \times \cdots \times m_n$ t_{SL}-time conflict combinations. In order to pursue all possible precedence orderings, the actual $\mathcal{RSV}$-diagram (D, t_{SL}) is *multiplied* by the number of the existing conflict combinations. Every conflict combination is assigned to one of the multiplied diagrams respectively. In every diagram, the assigned conflict activities are frozen and all the other t_{SL}-time conflict activities are moved behind each corresponding frozen activity respectively. We proceed with the step 4 for *every* diagram accompanied by A and T.
Step 4: **Deleting all t_{SL}-time direct scan-line activities from the actual activity term T:** If in the diagram (D, t_{SL}) t_{SL}-time direct scan-line activities exist, they surely have been frozen in the last step 3. Now all t_{SL}-time direct scan-line activities in (D, t_{SL}) are deleted from T of (D, t_{SL}). So T may become smaller.
Step 5: **Attaching further ground activities to the scan-line:** Further ground activities from the actual activity term T which can be attached to the scan-line are determined in order to place them. If in an actual diagram (D, t_{SL}) a scan-line activity $P(r, t)$ with $RE(P(r, t) > t_{SL}$ has been frozen, the resource r is being blocked until the time $RE(P(r, t))$. So, all further ground activities requiring *the t_{SL}-time blocked resource r* which have not yet been placed in the diagram and have no predecessor in T must wait until the scan-line has jumped to the time $RE(P(r, t))$. For $(D, t_{SL})(t_{SL} > 0)$ with an input activity term A and an actual activity term T, a ground activity $P(r, t)$ of T can be attached to the scan-line iff

1. P isn't from the diagram (D, t_{SL}),
2. in (D, t_{SL}) there exists no frozen activity $Q(r, l)$ for which $LE(Q) < t_{SL}$ and $RE(Q) > t_{SL}$ hold.

3. in T P has no predecessor.

Furthermore the steps 2, 3, 4 and 5 are recursively applied until all ground activities have been placed in the diagram and all activities in the diagram have been frozen so that T is empty and an active schedule is completed. Among all computed schedules, those that have the minimal project makespan are delivered as the optimal schedules for A.

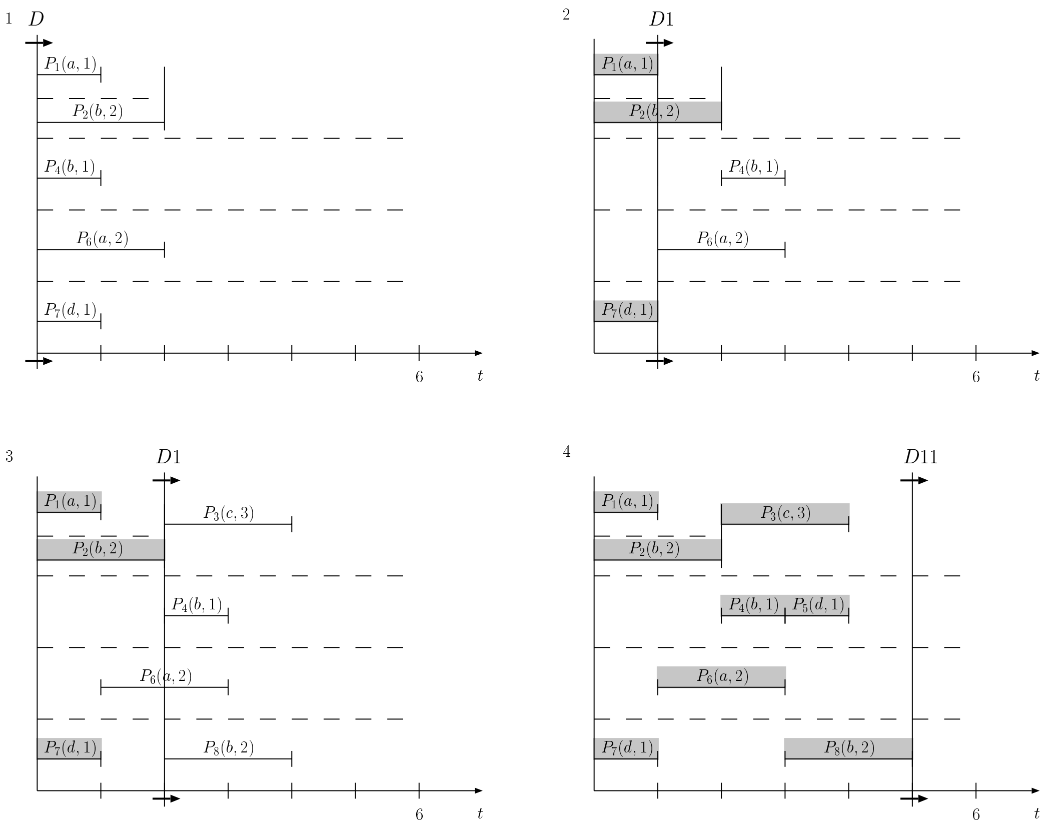

Figure3. $\mathcal{RSV}$-diagram-based calculation of active schedules for the reduced activity term of example 2

Example 3. In order to demonstrate that computing with the aid of $\mathcal{RSV}$-diagrams makes the algorithm easily understandable, we consider the second reduced activity term of example 2. In the beginning, the input activity term and the actual activity term are identical. There are start activities P_1, P_2, P_4, P_6 and P_7. The diagram 1 of figure 3 shows the resulting diagram after applying step 1, in which the scan-line time 0 has been assigned to these start activities

P_1, P_2, P_4, P_6 and P_7 as their start time. In step 2, the scan-line jumps into the next stopping time $t_{SL} = 1$ and we have the resulting diagram $(D, 1)$, in which there is one 1-time conflict free activity P_7 and two 1-time conflict resources a and b. So there exist four 1-time conflict combinations $[P_1, P_2], [P_1, P_4], [P_2, P_6]$ and $[P_4, P_6]$ altogether. The diagram $(D, 1)$ is multiplied 4 times, let these be $D1, \cdots, D4$ and $[P_1, P_2], [P_1, P_4], [P_2, P_6], [P_4, P_6]$ are assigned to $D1, \cdots, D4$ respectively. In every diagram, the 1-time conflict free activity P_7 and the assigned 1-time conflict activities are frozen and the other 1-time conflict activities are moved behind each corresponding frozen activity. Subsequently we proceed with the next step 4 in every diagram accompanied by the input activity term and the actual activity term.
If we pursue $(D1, 1)$ to which the combination $[P_1, P_2]$ is assigned, we have diagram 2 of figure 3 where P_1, P_2 and P_7 have been frozen and P_4 and P_6 have been moved behind P_2 and P_1 respectively. Now the two 1-time direct scan-line activities P_1 and P_7 in $(D1, 1)$ (diagram 2) have to be deleted from the actual activity term. After deleting both activities we have the following *new* actual activity term for $(D1, 1)$:

$$\begin{aligned}\textbf{pll}(&\textbf{seq}\ (\textbf{pll}\ P_2(b,2)),\ P_3(c,2)),\\ &(\textbf{seq}\ P_4(b,1), P_5(a,1)),\\ &P_6(a,2),\\ &(\textbf{seq}\ P_8(b,2))\end{aligned}$$

P_8 is here the unique activity which has no predecessor and isn't yet included in the diagram, but it requires the 1-time blocked resource b. 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 2, we have the diagram $(D1, 2)$ the diagram 3 of figure 3 shows, where the two further activities P_3 and P_8 newly have been attached to the scan-line. The information that P_7 corresponds to the immediate predecessor of P_8 could be read from the input activity term accompanied. So, P_8, for example, has been placed behind P_7 in the diagram. After applying the next step 2 to the diagram 3 we get $(D1, 3)$, in which there is a 3-time conflict resource b. $(D1, 3)$ is duplicated 2 times, let these be $(D11, 3)$ and $(D12, 3)$ where 3-time conflict activities P_4 and P_8 are assigned to $(D11, 3)$ and $(D12, 3)$ respectively. In $(D11, 3)$ the 3-time conflict free activity P_6 and the assigned activity P_4 are frozen etc. Finally, from the diagram $(D11, 3)$, one active schedule requiring the project makespan 5 is generated which the diagram 4 of figure 3 shows while from the diagram $(D12, 3)$, one active schedule requiring project makespan 6 is generated. For this example, there are 4 different optimal active schedules with project makespan 5 altogether.

4.3 Proving Correctness of $\mathcal{A}_{\mathcal{RSV}}$

A correctness proof for $\mathcal{A}_{\mathcal{RSV}}$ is given as follows:

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

Proof. We show the theorem through structural induction on the term construction of A.
Induction base: If A is a ground activity, the proof is trivial.
Induction step: In the beginning the scan-line is found at the time $t_{SL} = 0$. After applying the first step all start activities of A are attached to the scan-line. In the following the scan-line jumps to the next stopping time l. Let $G_1, \cdots, G_n$ be all l-time direct scan-line activities, i.e. it holds that $RE(G_1) = RE(G_2) = \cdots RE(G_n) = l$. Now, the following 2 different cases have to be distinguished:
Case 1: There is at least one activity G_i which corresponds to a l-time conflict-free activity. First, all l-time conflict-free activities are frozen and then the occurring resource conflicts are resolved. Here, let the diagram be multiplied to k diagrams $D_1, \cdots, D_k$ so that each l-time conflict combination is assigned to a diagram respectively. After resolving the resource conflicts, for every diagram all l-time direct scan-line activities are deleted from the actual activity term T respectively. So, in every diagram the corresponding actual activity term T becomes smaller since at least one l-time conflict-free activity G_i is deleted from T. Furthermore, $\mathcal{A}_{\mathcal{RSV}}$ is applied recursively to every diagram accompanied by the corresponding actual activity term T. By induction hypothesis, $\mathcal{A}_{\mathcal{RSV}}$ generates nonredundantly all active schedules for every diagram since every corresponding actual activity term T is smaller than A. Moreover, $\mathcal{A}_{\mathcal{RSV}}$ generates nonredundantly all active schedules for A since $D_1, \cdots, D_k$ are pairwise different. It is obviously true for the case $k = 0$, i.e. all $G_1, \cdots, G_n$ are l-time conflict-free too.
Case 2: Each activity G_i is involved in a l-time resource conflict, i.e. there is no l-time conflict-free activity. First the occurring resource conflicts are resolved. Let the diagram be multiplied to k diagrams $D_1, \cdots, D_k$ so that each l-time conflict combination is assigned to a diagram respectively. For any D_i the following two subcases have to be distinguished:
Case 2.1: There is at least one activity G_i which is frozen. This case is very similar to the case 1.
Case 2.2: None of the l-time direct scan-line activities $G_1, \cdots, G_n$ is frozen. Then, there are further l-time conflict activities which are frozen. Let these be $H_1, \cdots, H_m$ where $RE(H_j) > l$ for each j must hold. Eventually $H_1, \cdots, H_m$ will be deleted from the actual activity term T and in the result T will become structurally smaller. So, by induction hypothesis, $\mathcal{A}_{\mathcal{RSV}}$ generates nonredundantly all active schedules.
Consequently, for the case 2, $\mathcal{A}_{\mathcal{RSV}}$ generates nonredundantly all active schedules for A since $D_1, \cdots, D_k$ are pairwise different.

□

5 Summary and Future Work

The methods of description logics have been applied in order to formulate and solve a new general class of resource-constrained scheduling problems. Scheduling problems with variants are defined as activity terms of a concept language $\mathcal{RSV}$. The logic of $\mathcal{RSV}$ offered an effective approach for solving the $\mathcal{NP}$-complete

$\mathcal{RSV}$-problem. Furthermore, based on the language $\mathcal{RSV}$ a new diagram-based method for representing reduced activity terms of $\mathcal{RSV}$ has been introduced. The nonredundant generation of all active schedules for any reduced activity term could be described graphically using $\mathcal{RSV}$-diagrams, in whose center a scan-line principle stands.

The resource availability we discussed in this paper falls into category of type $n/1/1$, according to Holloway et al.'s [12] notation, where the most general category of type $n/n/n$ stands for multiple resource types, multiple units of resources and multiple number of resource types required by a ground activity. Until now, many models which deal with the classical $\mathcal{RCPS}$-problem and fall into the category of type $n/n/n$ (e. g. [17], [7], [8], [18]) have been introduced. In these models, a constant amount of each resource is assumed to be available throughout the duration of the project and to be also demanded by a ground activity throughout the duration of the ground activity.

Future work may investigate a generalization of resource availability of type $n/n/n$ for the language $\mathcal{RSV}$. Such general problems may be easily formulated by generalizing the syntax $P(r,t)$ to $P((r_1, r_2, \cdots, r_n), t)$, where n corresponds to the number of resource types and r_i ($i = 1, \cdots, n$) and $0 \leq r_i \leq b_i$ describes required units of resource type i by P. Here, each resource type i ($i = 1, \cdots, n$) is assumed to be available in a constant amount b_i throughout the duration of the project. Otherwise the three structural symbols (operators) '**seq**', '**xor**' and '**pll**' and the inductive rules for constructing activity terms may be applied unchanged.

References

1. A. Artale and Franconi E. A temporal description logic for reasoning about actions and plans. *J. Artificial Intelligence Research*, 9:463–506, 1998.
2. A. Artale and Franconi E. Temporal description logics. In *Handbook of Time and Reasoning in Artificial Intelligence.* MIT Press, 2000. forthcoming.
3. C. E. Bell and J. Han. A New Heuristic Solution Method in Resource-Constrained Project Scheduling. *Naval Research Logistics*, 38:315–331, 1991.
4. C. E. Bell and K. Park. Solving Resource-Constrained Project Scheduling Problems by $A*$ Search. *Naval Research Logistics Quarterly*, 37(1):61–84, 1990.
5. Fayez F. Boctor. Some efficient multi-heuristic procedures for resource-constrained project scheduling. *European Journal of Operational Research*, 49:3–13, 1990.
6. 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.
7. 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.
8. E. Demeulemeester and W. Herroelen. New Benchmark Results for the Resource-constrained Project Scheduling Problem. *Management Science*, 43(11):1485–1492, 1997.
9. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt:. The complexity of concept languages. *Information and Computation*, 134(1):1–58, 1997.

10. S. E. Elmaghraby. *Activity Networks: Project Planning and Control by Network Models.* Wiley, New York, 1977.
11. E. A. Elsayed and N. Z. Nasr. Heuristics for Resource Constrained Scheduling. *International Journal of Production Research*, 24(2):299–310, 1986.
12. C. A. Holloway, R. T. Nelson, and V. Suraphongschai. Comparison of a Multi-Pass Heuristic Decomposition Procedure with other Resource-Constrained Project Scheduling Procedures. *Management Science*, 25(9):862–872, 1979.
13. J. E. Jr. Kelly. *The Critical Path Method: Resource Planning and Scheduling, Ch 21 in Industrial Scheduling, Muth, J. F. AND Thompson, G. L. (eds.).* Prentice Hall, Englewood Cliffs, NJ, 1963.
14. M. M. Khattab and F. Choobineh. A New Approach for Project Scheduling with a Limited Resource. *International Journal of Production Research*, 29(1):185–198, 1991.
15. Pok-Son Kim. *Terminologische Sprachen zur Repräsentation und Lösung von ressourcenbeschränkten Ablaufplanungsproblemen mit Prozeßvarianten.* PhD thesis, Fachbereich Wirtschaftswissenschaften, Universität Frankfurt, 2001.
16. W. König, O. Wendt, and P. Rittgen. Das Wirtschaftsinformatik-Schwerpunktprogramm "verteilte DV-Systeme in der Betriebswirtschaft" der Deutschen Forschungsgemeinschaft -Frankfurt am Main, 1993-21 Bl. Technical Report 13, Institut für Wirtschaftsinformatik, 1993.
17. I. Kurtulus and E. W. Davis. Multi-Project Scheduling: Categorization of Heuristic Rules Performance. *Management Science*, 28(2):161–172, 1982.
18. 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.
19. O. Oguz and H. Bala. A Comparative Study of Computational Procedures for the Resource Constrained Project Scheduling Problem. *European Journal of Operational Research*, 72:406–416, 1994.
20. T. Ottmann and P. Widmayer. *Algorithmen und Datenstrukturen.* Wissenschaftsverlag, Mannheim/Wien/Zürich, 1990.
21. A. B. Pritsker, L. J. Watters, and P. M. Wolfe. Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16(1):93–108, September, 1969.
22. 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.
23. L. Schrage. Solving Resource-Constrained Network Problems by Implicit Enumeration-Nonpreemptive Case. *Operations Research*, 10:263–278, 1970.
24. 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.
25. F. B. Talbot. Resource-Constrained Project Scheduling with Time-Resource Trade-offs: The Nonpreemptive Case. *Management Science*, 28(10):1197–1210, 1982.
26. 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.
27. J. D. Wiest. *The Scheduling of Large Projects with Limited Resources.* PhD thesis, Carnegie Institute of Technology, 1963.
28. D. Wood. An Isothetic View of Computational Geometry. Technical Report CS-84-01, Department of Computer Science, University of Waterloo, Jan. 1984.