International Journal of Fuzzy Logic and Intelligent Systems, vol. 12, no. 1, March 2012, pp. 20-28
http://dx.doi.org/10.5391/IJFIS.2012.12.1.20 pISSN 1598-2645 eISSN 2093-744X

A Term-based Language for Resource-Constrained Project Scheduling and its Complexity Analysis

Arne Kutzner[1] **and Pok-Son Kim**[2*]

[1] **Department of Information Systems, Hanyang University, Seoul 133-791, Korea, Email: kutzner@hanyang.ac.kr**
[2] **Department of Mathematics, Kookmin University, Seoul 136-702, Korea, Email: pskim@kookmin.ac.kr**

Abstract

We define a language $\mathcal{RS}$, a subclass of the scheduling language $\mathcal{RSV}$ (*resource constrained project scheduling with variant processes*). $\mathcal{RS}$ involves the determination of the starting times for ground activities of a project satisfying precedence and resource constraints, in order to minimize the total project duration. In $\mathcal{RS}$ ground activities and two structural symbols (operators) '**seq**' and '**pll**' are used to construct activity-terms representing scheduling problems.
We consider three different variants for formalizing the $\mathcal{RS}$-scheduling problem, the *optimizing variant,* the *number variant* and the *decision variant*. Using the decision variant we show that the problem $\mathcal{RS}$ is $\mathcal{NP}$-complete. Further we show that the optimizing variant (or number variant) of the$\mathcal{RS}$-problem is computable in polynomial time iff the decision variant is computable in polynomial time.

Key words : Scheduling problem, description logics, complexity theory.

1. Introduction

Ever since the introduction of the pioneer works of Kelly [1] and Wiest [2] much has been reported about resource-constrained scheduling problems [3, 4, 5, 6, 7, 8]. Further, many various approaches for solving resource-constrained project scheduling ($\mathcal{RCPS}$) have been introduced. These are based on either heuristic algorithms [3, 4, 9, 10] or exact algorithms [11, 12, 13, 14]. Kim and Schmidt-Schauss suggested a new approach for representing and solving a general class of non-preemptive resource-constrained project scheduling problems in [15]. The new approach is to represent scheduling problems with variant processes as descriptions (activity terms) in a logic-based terminological language called $\mathcal{RSV}$ (***r**esource constrained project **s**cheduling with **v**ariant processes*). The language $\mathcal{RSV}$ involves the determination of the starting times for the activities of a project satisfying precedence and resource constraints in view of the possibility of generalization to "OR" activities in order to *minimize* the total project duration (See [15]). The vocabulary of $\mathcal{RSV}$ consists of a set of atomic activities $\{P_i(r(i), d(i)) | i = 1, \cdots, n (n \in \mathbb{N}), r(i) \in R, d(i) \in \mathbb{N}\}$, also called ground activities, and three structural symbols (operators) '**seq**', '**xor**' and '**pll**', where R is a finite set of resources. Each ground activity consists of a name P_i and two constants $r(i)$ and $d(i)$, where $r(i)$ and $d(i)$ denote P_i's associated resource and duration respectively. $r(i)$ and $d(i)$ are needed for satisfying the ground activity P_i. The operators '**seq**', '**pll**' and '**xor**' are used to construct expressions (activity-terms) and to specify the sequential processing of activity-terms, the possibility of parallel processing of activity-terms and the possibility of selecting an activity-term among several different alternative activity-terms respectively.
A model-theoretic semantics of activity-terms in $\mathcal{RSV}$ is given by an interpretation similar to that of description logics [16, 17, 15]. The use of semantic methods from description logics [16, 17, 15] is the key for understanding the meaning of activity terms. Description Logics belong to a research field in artificial intelligence and characteristics of them are a term language for concepts and other notions, a clean denotational semantics, and specific calculi (like subsumption) based on the semantics.
The most proposed methods for solving scheduling problems are 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 motivated 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 con-

Manuscript received Oct. 6, 2011; revised March 5, 2012; accepted March 10, 2012.
*Corresponding Author: Pok-Son Kim (kim@kookmin.ac.kr)
This work was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (2011-0004138) and Research Program 2011 of Kookmin University in Korea.
©The Korean Institute of Intelligent Systems. All rights reserved.

straints.
Further a calculus called $\mathcal{RSV}$-calculus and a diagram-based algorithm denoted by $\mathcal{A}_{\mathcal{RSV}}$ for solving the $\mathcal{RSV}$-scheduling problem have been introduced (See [15]).

In this paper we define a subclass of the language $\mathcal{RSV}$ that we call $\mathcal{RS}$. In $\mathcal{RS}$ only two operators '**seq**' and '**pll**' are used to construct activity-terms. The purpose of this paper is to investigate which time complexity is needed for solving $\mathcal{RS}$-problems. Although the $\mathcal{NP}$-completeness of the $\mathcal{RCPS}$-problem was already shown in [18], the proof of the $\mathcal{NP}$-completeness of such a problem-class as $\mathcal{RS}$ is still unknown. We will consider three different variants for formalizing the $\mathcal{RS}$-scheduling problem. These are *the optimizing variant, the number variant* and *the decision variant* of the $\mathcal{RS}$-problem. By using the decision variant we show that the problem $\mathcal{RS}$ belongs to the problem class $\mathcal{NP}$. Further, we will show that the well-known $\mathcal{NP}$-complete knapsack problem is polynomial reducible to the problem $\mathcal{RS}$. So, we can conclude the problem $\mathcal{RS}$ is $\mathcal{NP}$-complete. In addition, we will show that the optimizing variant (or number variant) of the $\mathcal{RS}$-problem is computable in polynomial time iff the decision variant is computable in polynomial time.

This paper is structured as follows. First, the language $\mathcal{RS}$ is defined in section 2. Then a calculus for the language $\mathcal{RS}$ is defined in section 3. The optimal solution algorithm is given in section 4. The time complexity of the problem $\mathcal{RS}$ is investigated in section 5. We will finish with some conclusion.

2. Formal Definition

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

The term-based scheduling language $\mathcal{RS}$ that is a subclass of $\mathcal{RSV}$ ([15]) is defined as follows:

Definition 2.1. The vocabulary of $\mathcal{RS}$ consists of a set of atomic activities $\{P_i(r(i), d(i)) | i = 1, \cdots, n(n \in \mathbb{N}), r(i) \in R, d(i) \in \mathbb{N}\}$, also called ground activities, and two structural symbols (operators) '**seq**' and '**pll**', where R is a finite set of resources. Each ground activity consists of a name P_i and two constants $r(i)$ and $d(i)$, where $r(i)$ and $d(i)$ denote P_i's associated resource and duration respectively. $r(i)$ and $d(i)$ are needed for satisfying the ground activity P_i.
The activity terms of $\mathcal{RS}$ 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

$$(\textbf{seq}\, A_1, A_2, \cdots, A_k),$$
$$(\textbf{pll}\, , A_2, \cdots, A_k),$$

are activity-terms.

The operators '**seq**' and '**pll**' are used to construct expressions (activity-terms) and to specify the sequential processing of activity-terms and the possibility of parallel processing of activity-terms respectively.

2.2 Schedules

Active schedules for $\mathcal{RS}$-activity-terms can be defined as follows:

For an 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 operator '**seq**'. It is generated on the set S of subterms of A as follows:

- $(\textbf{seq}\, B_1, \cdots, B_m) \in S \wedge i < j \Rightarrow B_i <_A B_j$.
- $B_i, B_j \in S \wedge B_i <_A B_j \Rightarrow B'_i <_A B'_j$ for every subterm B'_l of $B_l, l = i, j$.

Definition 2.2. Let A be an activity term and $g(A) = \{A_1, \cdots, A_n\}$. An active schedule for A is a set of starting times of ground activities $\{t_{A_i} \in \mathbb{N} | A_i \in g(A)\}$ 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 all $i \neq j$ and $t'_{A_j} < t_{A_j}$.

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

2.3 The Semantics of the Language $\mathcal{RS}$

Definition 2.3. The model-theoretic semantics of activity-terms in $\mathcal{RS}$ 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 $\mathcal{RS}$. 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.

2.4 The Scheduling Problem $\mathcal{RS}$

The objective is minimizing the project makespan. We call active schedules which have minimal project makespan *optimal active schedules*. So, we define the scheduling problem $\mathcal{RS}$ as follows:

For a given activity-term of $\mathcal{RS}$ an optimal active schedule has to be determined.

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

Two activity-terms are semantically equivalent, if the interpretations of the two activity-terms are identical. We take a part of the $\mathcal{RSV}$-calculus in [15] that we call $\mathcal{RS}$-calculus and it may be used to transform a $\mathcal{RS}$-expression A into a semantically equivalent $\mathcal{RS}$-expression B.

Definition 3.1. The $\mathcal{RS}$-calculus is defined as follows:

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 2 associative rules. Each associative rule describes that a subexpression combined by '**seq**' or '**pll**' which is an argument of the operator '**seq**' or '**pll**' respectively may be flattened:

$$\frac{(\textbf{seq}\, A_1, A_2, \cdots, A_k, (\textbf{seq}\, B_1, \cdots, B_l), A_{k+2}, \cdots, A_n)}{(\textbf{seq}\, A_1, A_2, \cdots, A_k, B_1, \cdots, B_l, A_{k+2}, \cdots, A_n)} \tag{1}$$

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

A rule in the form

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

is "correct" iff the interpretation of the upper expression A and the lower expression B are identical ($A^{\mathcal{I}} = B^{\mathcal{I}}$). The sets of all active schedules derived from the upper and lower expression of the *rule* (1) are 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 makes no change. Therefore, the following equation holds:

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

Hence the rule *Rule* (1) is correct. In a similar way we can show the rule (2) is correct as well.

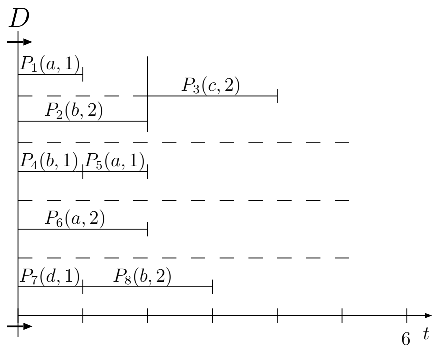

Figure 1: $\mathcal{RS}$-diagram with the scan-line on the stopping time $t_{SL} = 0$

4. Solution Algorithm of the $\mathcal{RS}$-Problem

It is known that for any activity-term A the set of active schedules derived from A is finite. All active schedules for any activity-term of $\mathcal{RS}$ can be calculated by the algorithm $\mathcal{A}_{\mathcal{RSV}}$ introduced in [15]. In this section we describe the algorithm $\mathcal{A}_{\mathcal{RSV}}$ briefly. Description of $\mathcal{A}_{\mathcal{RSV}}$ in detail can be found in [15]. $\mathcal{A}_{\mathcal{RSV}}$ is based on a new diagram-based method for representing activity-terms graphically. In this section we call the diagrams representing activity-terms in $\mathcal{RS}$ $\mathcal{RS}$-diagrams. Using $\mathcal{RS}$-diagrams explicit generation of all nonredundant active schedules for any activity-term can be illustrated graphically.
A $\mathcal{RS}$-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. The following example shows the representation of an activity-term by a $\mathcal{RS}$-diagram.

Example 4.1. The $\mathcal{RS}$-diagram of Figure 1 represents the following activity-term

$$\begin{aligned}\textbf{pll}\quad &(\textbf{seq}(\textbf{pll}P_1(a,1), P_2(b,2)), P_3(c,2)),\\ &(\textbf{seq}P_4(b,1), P_5(a,1)),\\ &P_6(a,2),\\ &\textbf{seq}P_7(d,1), P_8(b,2))\end{aligned}$$

In a $\mathcal{RS}$-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 the scan-line is found at the stopping time $t_{SL} = t$ in the $\mathcal{RS}$-diagram D. The scan-line is used for determining and resolving resource conflicts. 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.

In the beginning the scan-line is found at the time $t_{SL} = 0$ and the diagram is empty. Let A be given as any input activity-term.

Step 1: **Attaching start ground activities to the scan-line:** First all start ground activities of A 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.

Step 3: **Determining and resolving resource conflicts; Freezing all definitely placed ground activities:** In a $\mathcal{RS}$-diagram (D, t_{SL}) with $t_{SL} > 0$ a ground activity $P(r,t)$ is called a (t_{SL}-time) scan-line activity iff $\mathit{LE}(P(r,t)) < t_{SL}$ and $\mathit{RE}(P(r,t)) \geq t_{SL}$ hold. A scan-line activity $P(r,t)$ with $\mathit{RE}(P(r,t)) = t_{SL}$ is called a (t_{SL}-time) direct scan-line activity. If a direct scan-line activity $P(r,t)$ is a unique scan-line activity requiring the resource r, the begin and end time of such an activity $P(r,t)$ has been definitely determined. Therefore all such activities are frozen.

If in a $\mathcal{RS}$-diagram (D, t_{SL}) with $t_{SL} > 0$ ground activities $P_1(r,t_1), P_2(r,t_2), \cdots$ and $P_n(r,t_n) (n \geq 2)$ are all (t_{SL}-time) scan-line activities requiring the same resource r and there exists an activity $P_i(r,t_i)$, $i \in \{2,3,\cdots,n\}$ with $t_i = t_{SL}$, $P_1(r,t_1), P_2(r,t_2), \cdots$ 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. If there are t_{SL}-time scan-line activities involved in a (t_{SL}, r)-resource conflict, the 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 that a selected activity must not 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 them. There exist several different combinational possibilities for selecting activities. Such a combination is called *a conflict combination.* In order to pursue all conflict combinations, the $\mathcal{RS}$-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 selected activities are frozen and all the other t_{SL}-time conflict activities are moved behind each corresponding frozen activity respectively. We proceed then with the next step 4 for *every* diagram.

Step 4: **Deleting all t_{SL}-time direct scan-line activities from the activity-term A:** 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 A. So A may become smaller.

Step 5: **Attaching further ground activities to the scan-line:** Further ground activities from the activity-term A which can be attached to the scan-line are determined in order to place them. If in diagram (D, t_{SL}) a scan-line activity $P(r,t)$ with $\mathit{RE}(P(r,t) > t_{SL}$ has been frozen, the resource r is being blocked until the time $\mathit{RE}(P(r,t))$. 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 A must wait until the scan-line has jumped to the time $\mathit{RE}(P(r,t))$. For $(D, t_{SL})(t_{SL} > 0)$ with an activity-term A, a ground activity $P(r,t)$ of A 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 $\mathit{LE}(Q) < t_{SL}$ and $\mathit{RE}(Q) > t_{SL}$ hold.
3. in A 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 A becomes 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.

We already presented a correctness proof for this algorithm in [15].

5. Complexity

In this section we investigate which time complexity is needed for solving the $\mathcal{RS}$-problem. The investigation is based on the computation model “Turing machine”. The mathematical computation model “Turing machine” is often used in the field of complexity theory because this model allows to imagine the time duration for computation easily and intuitively [19, 20, 21, 22, 23].

5.1 $\mathcal{NP}$-completeness of the $\mathcal{RS}$-Problem.

A $\mathcal{NP}$-complete problem is considered as one of the “hardest” problems in the class $\mathcal{NP}$. The knapsack problem (*KP*, for short) is one of the well known $\mathcal{NP}$-complete problems. By showing the polynomial reducibility from the knapsack problem *KP* to the problem $\mathcal{RS}$ we prove the $\mathcal{NP}$-completeness of the problem $\mathcal{RS}$.

In $\mathcal{NP}$-completeness theory (complexity theory) decision variants of problems are used to analyze the time complexity of problems.

The $\mathcal{RS}$-Decision Problem

Let A be any expression in $\mathcal{RS}$. In the following we denote the total project duration of an active schedule P_A derived from A by $\mathit{tt}(P_A)$.

Let any expression $A \in \mathcal{RS}$ and a (minimal) time limit $t \in \mathbb{N}$ be given. The $\mathcal{RS}$-decision problem is:

- Does there exist *an optimal* and *active schedule* for A having the total project duration less than or equal to t ?

The Language $\mathcal{RS}^*$
The $\mathcal{RS}$-decision problem is to decide whether for a given expression $A \in \mathcal{RS}$ and for a given time limit $t \in \mathbb{N}$ an optimal and active schedule exists whose total duration is less than or equal to the time limit t.
The language $\mathcal{RS}^*$ consists of all pairs $(A,\ t)$ with $A \in \mathcal{RS}$ and $t \in \mathbb{N}$ such that for A there exists an optimal and active schedule P_A with $\mathit{tt}(P_A) \leq t$.

Knapsack Problem:
The knapsack problem is defined as follows:
Let $k_1, k_2, \cdots, k_n, r, n \in \mathbb{N}$ be given. The knapsack problem is:

- Does there exist an index set $I \subset \{1, 2, \cdots, n\}$ such that $\sum_{i \in I} k_i = r$?

The Language *KP*
The language KP consists of all triples $((k_1, k_2, \cdots, k_n), r, n)$ such that there exists an index set $I \subset \{1, 2, \cdots, n\}$ satisfying $\sum_{i \in I} k_i = r$ i.e. $KP = \{((k_1, k_2, \cdots, k_n), r, n) \mid \exists I \subset \{1, 2, \cdots, n\}$ such that $\sum_{i \in I} k_i = r\}$.

Now we show that $\mathcal{RS}^*$ can be recognized by a polynomial time nondeterministic algorithm, i. e. $\mathcal{RS}^* \in \mathcal{NP}$. Further we show that the language KP is polynomial reducible to $\mathcal{RS}^*$ ($KP \leq \mathcal{RS}^*$). By these both properties, it follows that the $\mathcal{RS}$-problem is $\mathcal{NP}$-complete.

Theorem 5.1. $\mathcal{RS}^*$ is in $\mathcal{NP}$.

Proof. A polynomial time nondeterministic Turing machine (abbreviated NT) accepting the strings in $\mathcal{RS}$ precisely can be constructed as follows: NT checks whether the input describes the correct encoding of a pair (A, t) with $A \in \mathcal{RS}$ and $t \in \mathbb{N}$. If the input is improper, NT stops and rejects it. Otherwise NT guesses an optimal active schedule for A in nondeterministic polynomial time. The nondeterministic guessing of an active schedule here corresponds to the optimal selection of a conflict combination each time in resolving resource conflicts similar to the algorithm $\mathcal{A}_{\mathcal{RSV}}$ (see section 4). In $\mathcal{RS}$-diagram computation of an active schedule for an activity-term of $\mathcal{RS}$ is calculated assigning a starting time to each ground activity of the activity-term. Therefore, it is obvious that guessing the optimal active schedule takes polynomial time. Afterwards NT calculates the minimal total duration of the guessed active schedule. If the total duration is less than or equal to t, then NT proceeds to the accepting state and halts; so the input string (A, t) has been accepted. If not, the input string (A, t) is rejected.
If (A, t) is in $\mathcal{RS}^*$, NT halts in an accepting state, since $\mathcal{RS}^*$ consists of all pairs $(A,\ t)$ with $A \in \mathcal{RS}$ and $t \in \mathbb{N}$

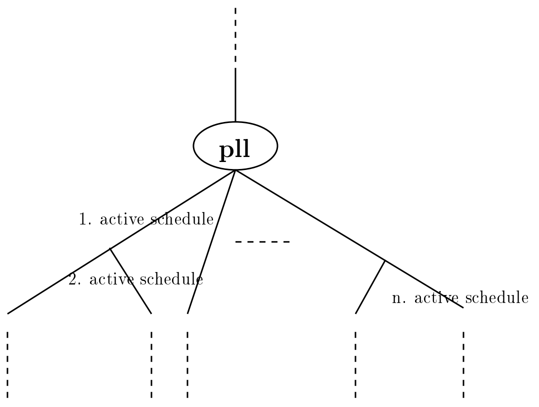

Figure 2: Nondeterminism with regard to the operator '**pll**'

such that for the expression A there exists an active schedule $P_{a_{ij}}$ with $\mathit{tt}(P_{a_{ij}}) \leq t$. If (A, t) is rejected, each optimal active schedule for A takes a total duration greater than t. □

Remark: The nondeterministic Turing machine NT constructed in theorem 5.1 can be considered as a variation of the algorithm $\mathcal{A}_{\mathcal{RSV}}$. NT is a nondeterministic Turing machine while the algorithm $\mathcal{A}_{\mathcal{RSV}}$ corresponds to a deterministic Turing machine. For a given input expression $A \in \mathcal{RS}$, NT guesses an optimal active schedule while $\mathcal{A}_{\mathcal{RSV}}$ computes all nonredundant active schedules for an expression. Then NT determines the corresponding optimal project duration in oder to compare the value with the time limit $t \in \mathbb{N}$.
Whenever in computation of the algorithm resource conflicts occur, the $\mathcal{RS}$-diagram is multiplied. This multiplication may be grasped as nondeterminism in NT. Whenever NT meets an expression combined by '**pll**', NT guesses an optimal schedule. If for an expression combined by '**pll**' n different active schedules would exist, at least one of the n schedules is surely an optimal schedule NT can guess. The figure 2 clarifies the nondeterminism with regard to the operator '**pll**':

Theorem 5.2. $KP \leq \mathcal{RS}^*$

Proof. Let a and b be two different resources and $K = \sum_{i=1}^{n} k_i$. For each weight k_i, $i = 1, 2, \cdots, n$ we take a ground activity $P_i(a, k_i)$, $i = 1, 2, \cdots, n$. We show the polynomial time reducibility $KP \leq \mathcal{RS}^*$ with the following assignment:

$$
\begin{aligned}
&((k_1, k_2, \cdots, k_n), r, n) \\
&\qquad \longmapsto_f \\
&((\textbf{pll}\, P_1(a, k_1), P_2(a, k_2), \cdots, P_n(a, k_n), \\
&\qquad (\textbf{seq}\, Q_1(b, r), Q_2(a, 1), Q_3(b, K - r))), K + 1)
\end{aligned}
$$

Now we have to show that $((k_1, k_2, \cdots, k_n), r, n) \in KP$ iff the assigned expression $((\textbf{pll}\ P_1(a, k_1), P_2(a, k_2), \cdots, P_n(a, k_n),$

$(\textbf{seq}\, Q_1(b,r), Q_2(a,1), Q_3(b,K-r))), K+1)$ is in $\mathcal{RS}^*$. Assume $((k_1,k_2,\cdots,k_n),r,n) \in KP$. Then there exists an index set $I \subset \{1,2,\cdots,n\}$ satisfying $\sum_{i\in I} k_i = r$. Therefore, while all P_is, $i \in I$ are carried out one by one, $Q_1(b,r)$ can be carried out in parallel with these. They need time duration r for their complete processing. Afterwards $Q_2(a,1)$ can be carried out. Because of a resource conflict $Q_2(a,1)$ can be carried out with no one of P_js, $j \in (\{1,2,\cdots,n\} - I)$ simultaneously. After processing Q_2, while all P_js, $j \in (\{1,2,\cdots,n\} - I)$ are carried out one by one, $Q_3(b,K-r)$ can be carried out simultaneously with these. Since $\sum_{j\in(\{1,2,\cdots,n\}-I)} k_j = K-r$, they exactly need time duration $K-r$. So we have an active schedule with the overall time duration $K+1$. Further, this active schedule is optimal, since all schedules need the overall time duration $K+1$ at least. Hence it holds $((\textbf{pll}\ P_1(a,k_1), P_2(a,k_2), \cdots, P_n(a,k_n), (\textbf{seq}\, Q_1(b,r), Q_2(a,1), Q_3(b,K-r))), K+1) \in \mathcal{RS}^*$. Conversely, assume $((k_1,k_2,\cdots,k_n),r,n) \notin KP$. Then there does not exist an index set $I \subset \{1,2,\cdots,n\}$ such that $\sum_{i\in I} k_i = r$. Without loss of generality, we assume while $P_1, P_2 \cdots$ and P_s are carried out one by one, $Q_1(b,r)$ is executed in parallel with these. Then it holds either $k_1 + k_2 + \cdots k_s < r$ or $k_1 + k_2 + \cdots k_s > r$. If $k_1+k_2+\cdots k_s < r$, then $k_{s+1}+\cdots+k_n > K-r$. Q_1 and Q_2 must be sequentially processed. Further Q_2, $P_{s+1}\cdots$ and P_n require the same resource a. Therefore all these have to be sequentially processed. So, they need a time duration greater than $K-r+1$. Hence all active schedules for $(\textbf{pll}\ P_1(a,k_1), P_2(a,k_2), \cdots, P_n(a,k_n), (\textbf{seq}\, Q_1(b,r), Q_2(a,1), Q(b,K-r)))$ need a total duration greater than $K+1$. In case of $k_1 + k_2 + \cdots k_s > r$: $P_1, P_2 \cdots, P_s$ and Q_2 have to be sequentially processed and these need a time duration greater than $r+1$. Thus all active schedules need a total duration greater than $K+1$ as well. Hence we have $((\textbf{pll}\ P_1(a,k_1), P_2(a,k_2), \cdots, P_n(a,k_n), (\textbf{seq}\, Q_1(b,r), Q_2(a,1), Q_3(b,K-r))), K+1) \notin \mathcal{RS}^*$. Trivially, it holds that $f \in \pi$ i. e. the time complexity needed for forming the assigned expressions is $O(n)$. □

The following examples clarify the polynomial time assignment and the $\mathcal{NP}$-completeness of $\mathcal{RS}^*$regarding theorem 5.2.

Example 5.3. Let the instance $((1,2,3,3,5),8,5) \in KP$ be given. Then the assigned expression is

$$((\textbf{pll}\ P_1(a,1), P_2(a,2), P_3(a,3), P_4(a,3), P_5(a,5), (\textbf{seq}\, Q_1(b,8), Q_2(a,1), Q_3(b,6))),\ 15)$$

For the above $\mathcal{RS}$-expression we can get an optimal active schedule with the makespan 15 as follows: First P_3 and P_5 are carried out one by one and Q_1 is carried out in parallel with P_3 and P_5. Subsequently Q_2 is executed. After executing Q_2, P_1, P_2 and P_4 are carried out one by one and Q_3 is performed in parallel with them. So, the expression is in $\mathcal{RS}^*$.

Let the instance $((4,5,6,7),3,4) \notin KP$ be given. Then the assigned expression is

$$((\textbf{pll}\ P_1(a,4), P_2(a,5), P_3(a,6), P_4(a,7), (\textbf{seq}\, Q_1(b,3), Q_2(a,1), Q_3(b,19))),\ 23)$$

$P_1, P_2 \cdots, P_4$ and Q_2 require the same resource a. So, they can not be processed simultaneously. We can consider the following optimal schedule. First P_1 and Q_1 are processed simultaneously and need the time duration 4. Afterward $Q_2$is processed. Then P_2, P_3 and P_4 are carried out in parallel with Q_3. This schedule needs the time duration 24. So we can conclude that for the above $\mathcal{RS}$-expression all active schedules require the total project duration 24 at least. Hence $((\textbf{pll}\ P_1(a,4), P_2(a,5), P_3(a,6), P_4(a,7), (\textbf{seq}\, Q_1(b,3), Q_2(a,1), Q_3(b,19))),\ 23) \notin \mathcal{RS}^*$.

Trivially, the complexity of the assignment from KP to $\mathcal{RS}^*$ is $O(n)$.

By Th. 5.1 and 5.2 we have the following:

Corollary 5.4. The problem $\mathcal{RS}$ is $\mathcal{NP}$-complete

5.2 Three Variants of the $\mathcal{RS}$-Problem

Depending on the output computed, three different variants of the $\mathcal{RS}$-scheduling problem may be distinguished. These three different variants are the optimizing variant, the number variant and the decision variant [23, 24]. We have already introduced the decision variant in section 5.1. In section 2 we have defined the $\mathcal{RS}$-scheduling problem as the optimizing variant. So, the optimizing and number variant are described as follows: Let an expression $A \in \mathcal{RS}$ be given. The optimizing variant of the $\mathcal{RS}$-problem is

- Compute an optimal active schedule for A.

The number variant of the $\mathcal{RS}$-problem is

- Compute the optimal total project duration for A.

5.3 $\mathcal{NP}$-completeness of the Optimizing and Number Variant

The $\mathcal{NP}$-completeness theory is a valuation model for time complexity of decision problems. Correspondingly we used the decision variant to prove the $\mathcal{NP}$-completeness of the $\mathcal{RS}$-scheduling problem in section 5.1. For many practical optimizing problems [23, 24] it is enough to deal with the decision variant because all three variants are equivalent regarding the complexity i. e. the optimizing variant (or number variant) is computable in polynomial time if and only if the decision variant is computable in polynomial time. Now we show this holds for the $\mathcal{RS}$-problem as well.

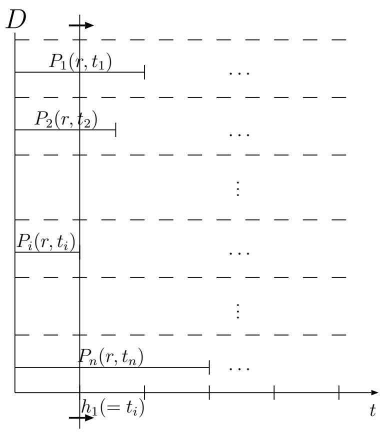

Figure 3: Computation of an optimal active schedule -$\mathcal{RS}$-diagram for $t(H) = h_1$

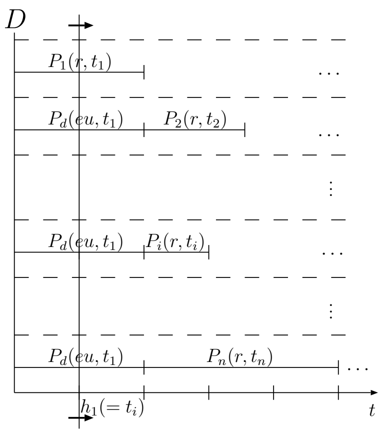

Figure 4: Computation of an optimal active schedule - Checking $P_1(r,t_1)$

Theorem 5.5. The number variant of the $\mathcal{RS}$-problem is efficiently computable if and only if the decision variant is efficiently computable.

Proof. Assume the number variant is efficiently computable. To solve the decision variant it is enough to check if the optimal total project duration given by the number variant is less than or equal to the time limit. Hence the decision variant is efficiently computable as well.

Conversely, the decision variant is efficiently computable. Let A be any $\mathcal{RS}$-expression and s be the sum of the durations of all ground activities of A. Obviously the optimal total project duration can not be greater than s. So, first we take $\lfloor \frac{s}{2} \rfloor$ as the minimal time limit and can get the answer "yes" or "no" using the decision variant. If we have the answer "yes", we take $\lfloor \frac{s}{4} \rfloor$ as the time limit once again. If "no", we take $\lfloor \frac{3s}{4} \rfloor$ as the time limit, and so on. We need $\log s$ iterations at most to calculate the exact optimal total project duration. So the number variant is efficiently computable. $\square$

Theorem 5.6. The optimizing variant of the $\mathcal{RS}$-problem is efficiently computable if and only if the number variant is efficiently computable.

Proof. Suppose the optimizing variant is efficiently computable. Then the number variant is efficiently computable as well, since the total project duration of an optimal schedule can be calculated efficiently.

Conversely, let G be an algorithm that solves the number variant efficiently. Let A be any input activity (a $\mathcal{RS}$-expression). First, we apply G to A to get the optimal total project duration α. Now, we begin deriving an optimal active schedule from A. Since it is enough to focus on **pll**-subexpressions and we can apply the calculus 2 (see section 3) to flatten **pll**-expressions, we assume A corresponds to a flattened **pll**-expression. To assign a starting time to each ground activity of A, we use the $\mathcal{RS}$-diagram. First A is represented as a $\mathcal{RS}$-diagram with stopping time $t(H) = 0$ of its scan-line. The scan line jumps to the next stopping time H with $t(H) = h_1$. For each h_1-time conflict resource, in order to find out which ground activity involved to the resource conflict has to be placed first, we check ground activities one by one as follows: Let $P_1(r,t_1), P_2(r,t_2), P_3(r,t_3), \cdots, P_n(r,t_n)$ be all h_1-time scan-line activities involved in (h_1, r)-resource conflict (see figure 3). We select the activity P_1 and "shift" all the remaining activities behind the selected activity P_1. We replace the empty places after shifting by a pseudo (dummy) ground activity $P_d(eu,t_1)$ (see figure 4). A pseudo (dummy) ground activity $P_d(eu,t_1)$ is such an activity that doesn't generate any resource conflict. Now we compute the optimal project duration for this changed activity term using G. Then we compare this value to α. If there is any increase, then P_1 is not in the right place. For at least one ground activity there must exist a successful check. For all other h_1-time conflict resources, the ground activities to be placed first are determined in the same way. Afterwards the scan-line jumps to the next stopping time H with $t(H) = h_2$. If there exist h_2-time conflict resources, the ground activities to be placed first are determined in the same way as before. By recursively applying this method, all resource conflicts can be resolved. Further the resulting conflict-free schedule is an optimal schedule, since its optimal project duration corresponds with α. The number of calls of the algorithm G is equal to $O(k^2)$ where k is the number of all ground activities in A. With the efficient number variant the optimizing variant is efficiently computable as well. $\square$

6. Conclusion

We introduced a term-based scheduling language $\mathcal{RS}$ that represents resource constrained project scheduling problems. Further we investigated the time complexity of the language $\mathcal{RS}$. We could show that the problem $\mathcal{RS}$ is $\mathcal{NP}$-complete. Further we could show the three variants of the $\mathcal{RS}$-problem, i.e. the optimizing variant, the number variant and the decision variant, are equivalent regarding their time complexity.

References

[1] J. E. J. Kelly, *The Critical Path Method: Resource Planning and Scheduling, Ch 21 in Industrial Scheduling, Muth, J. F. AND Thompson, G. L. (eds.).* Englewood Cliffs, NJ: Prentice Hall, 1963.

[2] J. D. Wiest, *The Scheduling of Large Projects with Limited Resources*. PhD thesis, Carnegie Institute of Technology, 1963.

[3] E. A. Elsayed and N. Z. Nasr, "Heuristics for resource constrained scheduling," *International Journal of Production Research*, vol. 24, no. 2, pp. 299–310, 1986.

[4] O. Oguz and H. Bala, "A comparative study of computational procedures for the resource constrained project scheduling problem," *European Journal of Operational Research*, vol. 72, pp. 406–416, 1994.

[5] E. Demeulemeester and W. Herroelen, "New benchmark results for the resource-constrained project scheduling problem," *Management Science*, vol. 43, no. 11, pp. 1485–1492, 1997.

[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*, vol. 107, pp. 272–288, 1998.

[7] C.-k. Kwon and K.-s. Oh, "Genetic algorithms based on maintaining a diversity of the population for job-shop scheduling problem," *Journal of Korean Institute of Intelligent Systems*, vol. 11, no. 3, pp. 191–199, 2001.

[8] K. M. Lee and K. M. Lee, "Task allocation and scheduling of multiagent systems with fuzzy task processing times," *Journal of Korean Institute of Intelligent Systems*, vol. 14, no. 3, pp. 324–329, 2004.

[9] A. Lim, H. Ma, B. Rodrigues, S. Tan, and F. Xiao, "New meta-heuristics for the resource-constrained project scheduling problem," *Flexible Services and Manufacturing Journal*, pp. 1–26, 2011.

[10] D. C. Paraskevopoulos, C. D. Tarantilis, and G. Ioannou, "Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm," *Expert Systems with Applications*, vol. 39, no. 4, pp. 3983–3994, 2012.

[11] E. Demeulemeester and W. Herroelen, "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem," *Management Science*, vol. 38, no. 12, pp. 1803–1818, 1992.

[12] 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*, vol. 44, no. 5, pp. 714–729, 1998.

[13] L. Bianco and M. Caramia, "Minimizing the completion time of a project under resource constraints and feeding precedence relations: A lagrangian relaxation based lower bound," *4OR*, vol. 9, no. 4, pp. 371–389, 2011.

[14] L. Bianco and M. Caramia, "An exact algorithm to minimize the makespan in project scheduling with scarce resources and generalized precedence relations," *European Journal of Operational Research*, vol. 219, no. 1, pp. 73–85, 2012.

[15] 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.

[16] M. Schmidt-Schauß and G. Smolka, "Attributive concept descriptions with unions and complements," Tech. Rep. SEKI Report SR-88-21, FB Informatik, Universität Kaiserslautern, D-6750, Germany, 1988.

[17] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt, "The complexity of concept languages," *Information and Computation*, vol. 134, no. 1, pp. 1–58, 1997.

[18] J. K. Lenstra and A. H. G. Rinnooy Kan, "Complexity of scheduling under precedence constraints," *Operations Research*, vol. 26, pp. 22–35, 1978.

[19] C. Posthoff and K. Schultz, *Grundkurs Theoretische Informatik*. Stuttgart; Leipzig: B. G. Teubner Verlagsgesellschaft, 1992.

[20] E. Börger, *Berechenbarkeit Komplexität Logik*. Vieweg, 1992.

[21] J. C. Martin, *Introduction to Languages and the Theory of Computation*. McGraw-Hill Inc., 1991.

[22] J. E. Hopcroft and J. D. Ullman, *Einfürung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*. Addison-Wesley, 1988.

[23] I. Wegener, *Theoretische Informatik*. Stuttgart: B. G. Teubner, 1993.

[24] I. Wegener, *Kompendium Theoretische Informatik-eine Ideensammlung*. Stuttgart: B. G. Teubner, 1996.

Pok-Son Kim
Professor of Kookmin University
Research Areas: Algorithm, Complexity theory, Scheduling problems.
E-mail : pskim@kookmin.ac.kr

Arne Kutzner
Professor of Hanyang University
Research Areas: Algorithm, Algorithm engineering, Programming languages.
E-mail : kutzner@hanyang.ac.kr