2003 International Symposium on Advanced Intelligent Systems
September 25-28, 2003, Jeju, Korea

Term-Based Scheduling Languages and their Comparison in View of the Expressivity

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

Abstract

The logic-based scheduling languages $\mathcal{RSV}$ and $\mathcal{RCPSV}$ may be used to represent and to solve a new general class of resource-constrained project scheduling with variants. $\mathcal{RSV}$ and $\mathcal{RCPSV}$ syntactically represent scheduling problems as descriptions (activity terms) being similar to concepts in a description logic. Though $\mathcal{RSV}$ and $\mathcal{RCPSV}$ have a different syntax, they are equally expressive. On the other hand, 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

Ever since the introduction of the pioneer works of Kelly [2]and Wiest (1963) [7] very much has been reported for the resource-constrained project scheduling problem [5], [6], [1], but in the classical methods for representing this NP-complete problem (for example, based on integer programming) it is generally difficult to read the flow structure and the content of a scheduling problem. This motivated us to use a term language for representing and solving scheduling problems. We have introduced a logic-based language called $\mathcal{RSV}$ [3] which syntactically represents resource-constrained scheduling problems as descriptions (activity terms) being similar to concepts in a description logic which emerged from KL-ONE-based, terminological knowledge representation systems of artificial intelligence [4]. The language $\mathcal{RSV}$ allows nested expressions using operators **pll**, **seq**, and **xor**. It generalizes previous approaches to scheduling with *variants* insofar as it permits **xor** 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 of $\mathcal{RSV}$ similar to propositional DNF-computation. The use of semantic methods from description logics is the key for understanding the meaning of compound activity terms. Characteristics of description logics are a term language for concepts and other notions, a clean denotational semantics, and specific calculi (like subsumption) based on the semantics. Based on these characteristics a diagram-based algorithm called $\mathcal{A}_{\mathcal{RSV}}$ for solving the $\mathcal{RSV}$-problem could be described which uses a scan-line principle to determine and resolve the occurring resource conflicts.
An additional logic-based terminological language called $\mathcal{RCPSV}$ which also may be used to model resource-constrained scheduling problems with variants also allows nested expressions using operators **hnet** and **xor** and covers activity-terms and semantics in a way best suited to the specific time and resource constraints similar to that of $\mathcal{RSV}$. Based on the semantics a calculus is defined which transforms an activity term into several separate subterms so that for each subterm all optimal schedules can be generated by using $\mathcal{A}_{\mathcal{RSV}}$.
Though $\mathcal{RSV}$ and $\mathcal{RCPSV}$ have a different syntax, we will show that they are equally expressive, i. e. each $\mathcal{RCPSV}$-expression can be represented as a $\mathcal{RSV}$-expression and vice versa. On the other hand, from the viewpoint of a syntactical representation ability of problems, $\mathcal{RCPSV}$ permits more compact representations than $\mathcal{RSV}$. We will present some scheduling examples for which the running time for translating into $\mathcal{RSV}$-term is more expensive than translating into $\mathcal{RCPSV}$-term.

2 The Scheduling Language $\mathcal{RSV}$

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 (nested expressions) and describing further constraints. Activity-terms 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

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

are activity-terms.

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 $\mathcal{A}_{\mathcal{RSV}}$. Among all these computed schedules, those that have the minimal value correspond then to the optimal schedules for the $\mathcal{RSV}$-term t.

3 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'). Based on this always existing simplification possibility the scheduling language $\mathcal{RCPSV}$ is defined as follows:
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 for specifying 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 and a scheduling network. 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 $\{n_{11}, n_{12}\} \cup \cdots \cup \{n_{k1}, n_{k2}\}$.

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 an interpretation function $\cdot^{\mathcal{I}}$. 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. Based on the semantics a calculus can be defined which transforms an activity term into a normalized activity term consisting of several separate reduced subterms so that for each subterm all optimal schedules can be generated by using $\mathcal{A}_{\mathcal{RSV}}$ (see [3]).

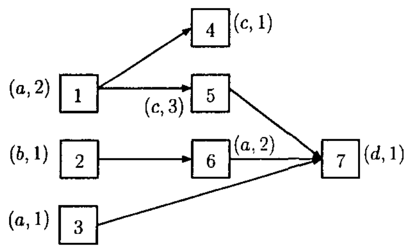

Figure 1: A network

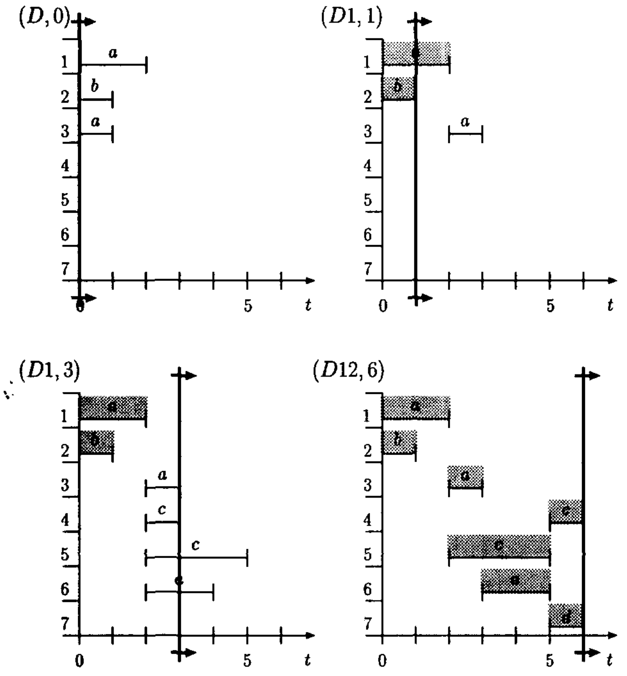

Figure 2: A $\mathcal{RCPSV}$-diagram based calculation

4 The Same, Powerful Expressivity of $\mathcal{RSV}$ and $\mathcal{RCPSV}$

We will prove 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 $\mathcal{A}_{\mathcal{RSV}}$ for a $\mathcal{RCPSV}$-term can be represented as a reduced $\mathcal{RSV}$-term $t(\sigma)$.

To show this, we consider an example. We take the active schedule $\sigma = (D12, 6)$ of figure 2 derived from the following $\mathcal{RCPSV}$-activity-term (see figure 1) using $\mathcal{A}_{\mathcal{RSV}}$.:

$$\begin{aligned}&\mathbf{hnet}[\mathit{let}\,1 = (1,a,2),\ 2 = (2,b,1),\ 3 = (3,a,1),\\ &4 = (4,c,1), 5 = (5,c,3), 6 = (6,a,2), 7 = (7,d,1);\\ &(1,4),(1,5),(2,6),(3,7),(5,7),(6,7)]\end{aligned}$$

With the aid of $\mathcal{A}_{\mathcal{RSV}}$, 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 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 2. 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 $t(\sigma_1)$:

$$\begin{array}{l}(\mathbf{pll}\ (\mathbf{seq}\,1),\\ \qquad (\mathbf{seq}\,2))\end{array} \tag{1}$$

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:

$$\begin{array}{l}(\mathbf{pll}\ (\mathbf{seq}\,3),\\ \qquad (\mathbf{seq}\,5))\end{array}$$

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

$$\begin{array}{l}(\mathbf{pll}\ (\mathbf{seq}\,1,\ \ (\mathbf{pll}\ (\mathbf{seq}\,3),\\ \qquad\qquad\qquad\qquad (\mathbf{seq}\,5))),\\ \qquad (\mathbf{seq}\,2))\end{array}$$

If, at the 3rd call, a further activity k with $LE(k) = 0$ had been frozen, k was combined by '**seq**' and then **seq** k was added to $t(\sigma_2)$ as an argument of the operator '**pll**' of (1). 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:

$$\begin{array}{l}(\mathbf{pll}\ (\mathbf{seq}\,1,\ \ (\mathbf{pll}\ (\mathbf{seq}\,3,6),\\ \qquad\qquad\qquad\qquad (\mathbf{seq}\,5, (\mathbf{pll}\,4,7)))),\\ \qquad (\mathbf{seq}\,2))\end{array}$$

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

Lemma 4.1. *For a network (X, P) of a given $\mathcal{RCPSV}$-expression, let σ be any complete active schedule delivered through $\mathcal{A}_{\mathcal{RSV}}$ and n be the number of recursive calls of*

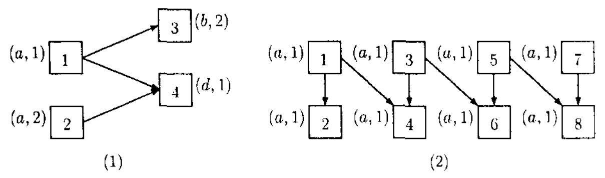

Figure 3: $\mathcal{RCPS}$-problems

the step 3 "Freezing all definitely placed ground activities" *of $\mathcal{A}_{\mathcal{RSV}}$ 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**' forces the activities $d_1, d_2, \cdots, d_m$ frozen at the $(i+1)$th call to have exactly the same schedule time for carrying out just as in σ_{i+1}. □

Theorem 4.1. *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 $\mathcal{A}_{\mathcal{RSV}}$. In Lemma 4.1 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 4.1 the expression $(\mathbf{xor}\, t(\tau_1), \cdots, t(\tau_n))$ presumably can have an exponentially larger space demand than the activity-term s.

We consider the $\mathcal{RCPS}$-problem represented by (1) of figure 3. This problem can be represented by the $\mathcal{RCPSV}$-activity-term (2) in a *compact* form while as a $\mathcal{RSV}$-activity-term, first, for the problem all nonredundant active schedules must be computed and then, with the aid of the construction rule described above, a corresponding $\mathcal{RSV}$-expression such as (3) can be described.

$$\begin{aligned}(\mathbf{hnet}[let\, 1 = (1, a, 1), 2 = (2, a, 2), 3 = (3, b, 2),\\ 4 = (4, d, 1); (1,3), (1,4), (2,4)])\end{aligned} \quad (2)$$

$$\begin{aligned}(\mathbf{xor}\quad &(\mathbf{seq}\,(1, a, 1), (\mathbf{pll}\,(3, b, 2), (\mathbf{seq}(2, a, 2), (4, d, 1))),\\ &(\mathbf{seq}\,(2, a, 2), (1, a, 1), (\mathbf{pll}\,(3, b, 2), (4, d, 1))))\end{aligned} \quad (3)$$

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, (2) of figure 3 presumably shows such a problem.

5 Summary

Though $\mathcal{RSV}$ and $\mathcal{RCPSV}$ have a different syntax, we could 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 $\mathcal{RSV}$. We have presented some scheduling examples for which the running time for translating into $\mathcal{RSV}$-term is more expensive than translating into $\mathcal{RCPSV}$-term.

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] 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.

[3] 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.

[4] 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.

[5] L. Schrage. Solving resource-constrained network problems by implicit enumeration, preemptive case. *Operations Research*, 20(3):668–677, 1972.

[6] 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.

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