Description Logics for Solving Scheduling Problems

Pok-Son Kim
Kookmin University, College of Natural Sciences
Department of Mathematics
Seoul 136-702, Korea
Abstract

We introduce new methods for representing and solving general classes of non-preemptive resource-constrained project scheduling problems that minimize the project makespan. These methods are based on a new approach to represent scheduling problems as descriptions (activity-terms) of terminological languages called $RSV$ and $RCPSV$, which allow nested expressions using $pll$, $seq$, $xor$ and $hnet$, $xor$ respectively. The activity-terms of $RSV$ and $RCPSV$ are similar to concepts in a description logic. The languages $RSV$ and $RCPSV$ generalize previous approaches to scheduling with variants insofar as it permits $xor$’s not only of atomic activities but also of arbitrary activity-terms. Specific semantics that assign their set of active schedules to activity-terms show correctness of calculuses normalizing activity-terms of $RSV$ and $RCPSV$ respectively similar to propositional DNF-computation.
Overview

1. $\text{RSV}$- and $\text{RCPSV}$-Scheduling Problems

2. The Scheduling Language $\text{RSV}$

3. A Calculus for $\text{RSV}$

4. The Algorithm $\mathcal{A}_{\text{RSV}}$

5. The Scheduling Language $\text{RCPSV}$

6. A Calculus for $\text{RCPSV}$
$\mathcal{RCSV}$-Problems

$\mathcal{RC}$: resource constrained

$\mathcal{PS}$: project scheduling

$\mathcal{V}$: with variants
Graph, Example

represented as ANDORI-graph

- variant processes
- precedence constraints
- parallel processing
- resource constraints
Methods for formalizing and Solving \( \text{RCP\textsc{SV}} \)-Problems

- Definition of the logic-based Scheduling languages \( \text{RSV} \) and \( \text{RCP\textsc{SV}} \).

- Diagram-based Solution Algorithms
  - \( \text{RSV} \)-Diagram Calculation
  - \( \text{RCP\textsc{SV}} \)-Diagram Calculation
The Scheduling Language \( RSV \) - Definition

The Syntax of the Language \( RSV \)

- **ground (atomic) activities**
  
  \[ P(r, t) \]
  
  \( P \) : predicatesymbol
  \( r \) : resource (human, machine, asset, \( \cdots \))
  \( t \in N \) : duration

- **operators**
  
  - **seq**: sequential processing of an activity term or activity terms
  - **xor**: alternative activity terms
  - **pll**: parallel processing of activity terms
The Scheduling Language \( \mathcal{RSV} \) - Definition

- activity terms
  1. Each ground activity is an activity term
  2. \( P_1, P_2, \ldots, P_k \): activity terms, then
     \( \text{seq} \ P_1, \ldots, P_k \),
     \( \text{xor} \ P_1, \ldots, P_k \),
     \( \text{pll} \ P_1, \ldots, P_k \) are also activity terms.

Example 1:

\[
\text{seq} \ (\text{xor} \ P_1(r_1, 15), \ (\text{pll} \ P_2(r_2, 2), \ P_3(r_2, 2))) \\
(\text{pll} \ P_4(r_3, 10), \ P_5(r_4, 12))
\]
Activity-terms; Example 2

\[
\text{seq} M_1, (\text{pll} (\text{seq} M_{31}, M_{32}), (\text{xor} (\text{seq} M_{211}, M_{212}), M_{22})), M_4
\]
Reduced Activity Terms - Definition

$A$, $B$: $RSV$-expressions

$B$ is a reduced activity subterm of $A$, if $B$ can be derived from $A$ by repeatedly replacing subterms of the form $(xor \, C_1, \ldots, C_n)$ by exactly one $C_i$ ($i = 1, \ldots$ or $n$) so that $B$ is $xor$-free.

Example:
The 2 reduced activity terms of the previous activity-term:

\[
(\text{seq} \, \, P_1(r_1, 15), \\
(\text{pll} \, P_4(r_3, 10), P_5(r_4, 12)))
\]

\[
(\text{seq} \, \, (\text{pll} \, P_2(r_2, 2), P_3(r_2, 2)), \\
(\text{pll} \, P_4(r_3, 10), P_5(r_4, 12)))
\]
Active Schedules - Definition

\( A \): a reduced activity term

\( g(A) = \{A_1, \ldots, A_n\} \): the set consisting of all ground activities occurring in \( A \)

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,
- The resource constraints are satisfied and
- No ground activity can be started earlier without changing other start times.
Semantics of the Language $\mathcal{RSV}$ - Definition

**Interpretation** $\mathcal{I} = (\mathcal{D}, \cdot \mathcal{I})$

$\mathcal{D}$: the set consisting of all active schedules derived from activity terms in $\mathcal{RSV}$

$\cdot \mathcal{I}$: the interpretation function of $\mathcal{I}$

![Diagram](image-url)
The \( RSV \)-Calculus

\[
\frac{(\text{seq } A_1, A_2, \cdots, A_k, (\text{seq } B_1, B_2, \cdots, B_l), A_{k+2}, A_{k+3}, \cdots, A_n)}{(\text{seq } A_1, A_2, \cdots, A_k, B_1, B_2, \cdots, B_l, A_{k+2}, A_{k+3}, \cdots, A_n)}
\]

(1)

\[
\frac{(\text{xor } A_1, A_2, \cdots, A_k, (\text{xor } B_1, B_2, \cdots, B_l), A_{k+2}, A_{k+3}, \cdots, A_n)}{(\text{xor } A_1, A_2, \cdots, A_k, B_1, B_2, \cdots, B_l, A_{k+2}, A_{k+3}, \cdots, A_n)}
\]

(2)

\[
\frac{(\text{pll } A_1, A_2, \cdots, A_k, (\text{pll } B_1, B_2, \cdots, B_l), A_{k+2}, A_{k+3}, \cdots, A_n)}{(\text{pll } A_1, A_2, \cdots, A_k, B_1, B_2, \cdots, B_l, A_{k+2}, A_{k+3}, \cdots, A_n)}
\]

(3)
$\begin{align}
\text{seq} A_1, A_2, \cdots, A_k, (\text{xor} B_1, B_2, \cdots, B_l), A_{k+2}, A_{k+3}, \cdots, A_n) \\
\text{xor} (\text{seq} A_1, A_2, \cdots, A_k, B_1, A_{k+2}, A_{k+3}, \cdots, A_n), \\
(\text{seq} A_1, A_2, \cdots, A_k, B_2, A_{k+2}, A_{k+3}, \cdots, A_n), \\
\vdots \\
(\text{seq} A_1, A_2, \cdots, A_k, B_l, A_{k+2}, A_{k+3}, \cdots, A_n)) \\
\end{align}$

$\begin{align}
\text{pll} A_1, A_2, \cdots, A_k, (\text{xor} B_1, B_2, \cdots, B_l), A_{k+2}, A_{k+3}, \cdots, A_n) \\
\text{xor} ((\text{pll} A_1, A_2, \cdots, A_k, B_1, A_{k+2}, A_{k+3}, \cdots, A_n), \\
(\text{pll} A_1, A_2, \cdots, A_k, B_2, A_{k+2}, A_{k+3}, \cdots, A_n), \\
\vdots \\
(\text{pll} A_1, A_2, \cdots, A_k, B_l, A_{k+2}, A_{k+3}, \cdots, A_n)) \\
\end{align}$
Correctness of the $\mathcal{RSV}$-Calculus

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

*Proof.* idea: 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 ($A^I = B^I$). We can show this for each of the 5 rules.
**Theorem** For any activity description $A$ of $RSV$ 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. idea:**

- All derivations terminate in a normalized expression (This is the case when no further $RSV$-rules can be applied.).

- In every expression containing ‘xor’-operator, it can be shifted to the topmost position.
Example

Calculus that may be used to transform any $RS\mathcal{V}$-expression $A$ into a semantically equivalent normalized $RS\mathcal{V}$-expression $B$:

\[
(xor \ (pll \ (seq \ P_3(c,15), \ P_4(c,16)), \ P_5(c,3)), \\
(pll \ (seq \ P_3(c,15), \ P_4(c,16)), \ P_6(d,5)))
\]

normalized by the $RS\mathcal{V}$-calculus
Graphical Representation of a Scheduling Equation

\[ A \models B \]
A $\mathcal{RSV}$-Diagram

Graphical Representation of a $\mathcal{RSV}$-Term

$$\text{pll (seq } P_1(a, 1), (\text{pll } P_2(b, 1), P_3(d, 2)), P_4(c, 3))$$

(seq $P_5(d, 2), P_6(a, 1)$)

$P_7(b, 3)$
Solution Algorithm $A_{RSV}$ Based on a Scan-Line Principle

- Attaching start ground activities to the scan-line

Recursive application, only if some unfrozen ground activity exists:

- Moving the scan-line

- Determining and resolving resource conflicts; Freezing all definitely placed ground activities

- Deleting all $t_{SL}$-time direct scan-line activities from the actual activity term

- Attaching further ground activities to the scan-line
Figure 1: \textit{RSV}-Diagram-Based Calculation of Active Schedules
Proving Correctness of $A_{RSV}$

**Theorem:** For any given reduced $RSV$-activity term $A$, $A_{RSV}$ generates nonredundantly all active schedules which may be derived from $A$. 
Complexity of the language $RSV$

$RSV$ is $\mathcal{NP}$-complete

The language $RSV^*$

$RSV^*$ consists of all Pairs $(X, t)$ with $X \in RSV$ and $t \in \mathbb{N}_0$, where there is an active schedule $P_{aij}$ with $t(P_{aij}) \leq t$ for the term $X$.

Theorem: $RSV^* \in \mathcal{NP}$.

Theorem: $SE \leq RSV^*$
The Scheduling Language $\textit{RCP\textsc{sv}}$ - Definition

The Syntax of the Language $\textit{RCP\textsc{sv}}$

- **h-ground activities**

  \[\{(0, eu, 0)\} \cup \{(i, r(i), d(i))|i = 1, \cdots, n, r(i) \in R, d(i) \in \mathbb{N}_0\}\]

  $eu$ : a dummy-resource

- **operators**
  
  - **xor**: alternative activity terms
  
  - **hnet**: structural arrangement of activity terms; The structural arrangement of activity terms of the operator ‘hnet’ always has to represent a directed simple and acyclic graph (a scheduling network).

  \[\text{hnet}[\text{let } t_1 = N_1, \cdots, t_k = N_k;(t_{11}, t_{12}), \cdots, (t_{j1}, t_{j2})]\]
The Scheduling Language \textit{RCPSV} - Definition

- h-activity terms

1. Each h-ground activity is a h-activity term.
2. $N_1, N_2, \cdots, N_k$: h-activity terms, then
   \( \text{xor}\ N_1, N_2, \cdots, N_k \) and
   all expressions

\[ \text{hnet[let } t_1 = N_1, \cdots, t_k = N_k; (t_{11}, t_{12}), \cdots, (t_{j1}, t_{j2}) \text{]} } \]

are h-activity terms, where $t_1, t_2, \cdots, t_k$ correspond to different names (constants) and \([(t_{11}, t_{12}), \cdots, (t_{j1}, t_{j2})]\) represents a scheduling network on \(\{t_1, \cdots, t_k\}\).
Example

\[
\text{\textbf{hnet}}[\text{\textbf{let}} \ t_1 = (1, a, 2), t_2 = (\text{xor}(2, b, 1), (3, c, 2)); (t_1, t_2)], \\
\text{\textbf{hnet}}[\text{\textbf{let}} \ k_1 = (0, eu, 0), k_2 = (4, d, 2), k_3 = (5, d, 2); (k_1, k_2), (k_1, k_3)]
\]
Semantics of the Language $\mathcal{RCPSV}$ - Definition

**Interpretation** $\mathcal{I} = (\mathcal{D}, \cdot \mathcal{I})$

$\mathcal{D}$: the set consisting of all active schedules derived from activity terms in $\mathcal{RCPSV}$

$\cdot \mathcal{I}$: the interpretation function of $\mathcal{I}$
The *RCP$SV$*-Calculus

Calculus that may be used to transform any *RCP$SV$*-expression $t$ into a semantically equivalent *normalized* *RCP$SV$*-expression $s$.

$t_1, t_2, \cdots, t_k, s_1, \cdots, s_l, t_{k+2}, \cdots, t_n$: activity-terms

The calculus has the associative rule (6) and distributive rules (7) and (8).
\[
\frac{(\text{xor } t_1, t_2, \ldots, t_k, (\text{xor } s_1, s_2, \ldots, s_l), t_{k+2}, t_{k+3}, \ldots, t_n)}{(\text{xor } t_1, t_2, \ldots, t_k, s_1, s_2, \ldots, s_l, t_{k+2}, t_{k+3}, \ldots, t_n)}
\]  

(\text{hnet}[\text{let } n_1 = t_1, \ldots, n_{k+1} = (\text{xor } s_1, s_2, \ldots, s_l), \ldots, n_n = t_n; (n_{11}, n_{12}), \ldots, (n_{h_1}, n_{k+1}), \ldots, (n_{j_1}, n_{j_2})])

(\text{hnet}[\text{let } n_1 = t_1, \ldots, n_{k+1} = s_1, \ldots, n_n = t_n; (n_{11}, n_{12}), \ldots, (n_{h_1}, n_{k+1}), \ldots, (n_{j_1}, n_{j_2})],
\text{hnet}[\text{let } n_1 = t_1, \ldots, n_{k+1} = s_l, \ldots, n_n = t_n; (n_{11}, n_{12}), \ldots, (n_{h_1}, n_{k+1}), \ldots, (n_{j_1}, n_{j_2})])

(\text{hnet}[\text{let } n_1 = t_1, \ldots, n_{k+1} = (\text{xor } s_1, s_2, \ldots, s_l), \ldots, n_n = t_n; (n_{11}, n_{12}), \ldots, (n_{k+1}, n_{h_2}), \ldots, (n_{j_1}, n_{j_2})])

(\text{hnet}[\text{let } n_1 = t_1, \ldots, n_{k+1} = s_1, \ldots, n_n = t_n; (n_{11}, n_{12}), \ldots, (n_{k+1}, n_{h_2}), \ldots, (n_{j_1}, n_{j_2})],
\text{hnet}[\text{let } n_1 = t_1, \ldots, n_{k+1} = s_l, \ldots, n_n = t_n; (n_{11}, n_{12}), \ldots, (n_{k+1}, n_{h_2}), \ldots, (n_{j_1}, n_{j_2})])

\text{Rules (6), (7), (8)}
Proving correctness of the $\mathcal{RCP SV}$-calculus

Lemma The $\mathcal{RCP SV}$-calculus is a correct calculus.

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

Solution Algorithm $A_{\mathcal{RCP SV}}$ Based on a Scan-Line Principle
Conclusion

- The Logic-based Languages $\mathcal{RSV}$ and $\mathcal{RCP SV}$ for Representing and Solving $\mathcal{RCP SV}$-Scheduling Problems

- Diagram-based Solution Algorithms