数理解析研究所講究録
918 巻 1995 年 48-57

Analysis of Completeness of Laboratory Assignment Algorithm by Rewrite Systems Theory

Masato NOTO, Masahito KURIHARA and Azuma OHUCHI

Institute of Systems and Information Engineering
Faculty of Engineering, Hokkaido University
Sapporo, 060 Japan
E-mail: {noto,kurihara,ohuchi}@huie.hokudai.ac.jp

Abstract

A laboratory assignment algorithm is a procedure that assigns each of the $m$ students to one of the $n$ laboratories. In this paper, we prove the *completeness* (i.e., *termination* and *confluence*) of the algorithm by using *abstract reduction systems* theory. Termination guarantees that the computation will not proceed indefinitely, and confluence guarantees that the computational result is unique even in the presence of indeterminacy.

1 Introduction

Laboratory assignment requires each of the given students to be assigned to one of the given laboratories of the university for graduation research. In this paper, we consider an algorithm derived from a real procedure by simplification and abstraction to make theoretical consideration easy.

The algorithm takes the following data as input:

(1) the lists of wishes of the students:

Each student submits a list of his/her wishes for laboratories. This is a list of all the laboratories ordered in accordance with his/her wishes to be assigned to.

(2) the priority on the students for each laboratory:

Each laboratory defines its own priority on the students.

(3) seating capacity:

Each laboratory has its own seating capacity. The sum of the seating capacities should be equal to the number of the students.

Let $S = \{S_1, \ldots, S_m\}$ be a set of *students* and $L = \{L_1, \ldots, L_n\}$ be a set of *laboratories.* Associated with each laboratory L_j are a natural number C_j, called the *capacity*, and a strict total ordering $\succ_j$ on S, called the *preference of the laboratory.* On the other hand, each student S_i is associated with a strict total ordering $\sqsubset_i$ on L, called the *preference of the student.* Informally, $S_a \succ_j S_b$ specifies that the laboratory L_j prefers S_b to S_a, and $L_a \sqsubset_i L_b$ means that the student S_i prefers L_a to L_b. The following procedure computes a mapping α that assigns each student S_i to a laboratory $\alpha(S_i)$ according the preferences and the capacity constraints $|\alpha^{-1}(L_j)| \leq C_j$ of all the laboratories L_j.

The algorithm is as follows:

1. For all i, $1 \leq i \leq m$, let H_i be the list of laboratories arranged in the ascending order of $\sqsubset_i$, i.e., $H_i = (L_{j_1} L_{j_2} \ldots L_{j_n})$ with $L_{j_1} \sqsubset_i L_{j_2} \sqsubset_i \ldots \sqsubset_i L_{j_n}$. (We call H_i the current preference list of the student S_i. When the first element of H_i is L_j, we say that S_i is temporarily assigned to L_j. Thus after the first step of the procedure, all the students are temporarily assigned to the most preferred laboratories. Given H_i for all i, A_j denotes the set of the students temporarily assigned to L_j.)

2. Repeat the following steps (**2.1** – **2.2**) while $(\exists j, 1 \leq j \leq n)|A_j| > C_j$, i.e., while there is a laboratory L_j such that the number of the students temporarily assigned to it exceeds its capacity.

- **2.1** Arbitrarily select a laboratory L_j satisfying $|A_j| > C_j$.
- **2.2** Let R be the set of $|A_j| - C_j$ students taken from A_j in the descending order of $\succ_j$. (Consequently, if $S_a \in R$ and $S_b \in A_j \backslash R$ then $S_a \succ_j S_b$.) For all students S_i in R, update H_i by removing its first element. (As the result, the students get temporarily assigned to other laboratories.)

3. Let the current temporary assignment be the final assignment, i.e., for all i, let $\alpha(S_i)$ be the first element of H_i.

The step **2.1** is called the *selection* of a laboratory and the step **2.2** is called the *adjustment* to the selected laboratory. Note that even after the selection of (and adjustment to) L_j, it is possible for the same L_j to be selected again in the future computation, because later adjustment to other laboratories could result in the increase of the students temporarily assigned to L_j. Also note how the preference $\succ_j$ of the selected laboratory is used in the adjustment.

The most crucial aspect of the procedure is its indeterminacy resulted from the arbitrary selection in step **2.1**. The procedure would be considered inadequate in practice if the solution (computational result) were not unique because of the dependence upon the selection. In that case, it would be difficult to design an appropriate selection strategy. Thus, the computational result should be unique, independent of the selection strategies. In this paper, we show that this uniqueness actually holds for this procedure, by proving the completeness (i.e., strong normalization and confluence) of the abstract reduction system derived from the procedure.

In section 2, we review the theory of abstract reduction systems. An *abstract reduction system* (ARS) is a pair $\mathcal{A} = \langle A, \rightarrow \rangle$ consisting of a set A and a binary relation $\rightarrow \subseteq A \times A$. The transitive closure of $\rightarrow$ is denoted by $\rightarrow^+$. The reflexive-transitive closure of $\rightarrow$ is denoted by $\twoheadrightarrow$. An element $a \in A$ is a *normal form* if there exists no element $b \in A$ with $a \rightarrow b$. ARS's properties are defined as follows:

- An ARS $\mathcal{A}$ is *strongly normalizing* (SN) if there are no infinite sequences $a_1 \rightarrow a_2 \rightarrow \cdots$ of elements of A.
- An ARS $\mathcal{A}$ is *confluent* or has the *Church-Rosser property* (CR) if for all elements $a, b, c \in A$ with $b \twoheadleftarrow a \twoheadrightarrow c$, there exists some element $d \in A$ with $b \twoheadrightarrow d \twoheadleftarrow c$.
- An ARS $\mathcal{A}$ is *locally confluent* or has the *weakly Church-Rosser property* (WCR) if for all elements $a, b, c \in A$ with $b \leftarrow a \rightarrow c$, there exists some element $d \in A$ with $b \twoheadrightarrow d \twoheadleftarrow c$.
- An ARS $\mathcal{A}$ has the *unique normal form property with respect to reduction* (UN$^{\rightarrow}$) if for all elements $a, b, c \in A$ with $b \twoheadleftarrow a \twoheadrightarrow c$, b and c being normal forms of $\mathcal{A}$, we have $b = c$.

Our proof of the uniqueness will be based on the following important propositions:

- Every CR system has the UN$^{\rightarrow}$ property.
- Every system that is both SN and WCR is CR. (Newman's lemma)

We call an ARS *complete* if it is SN and CR.

In section 3, we formalize our procedure as an ARS $\langle \Sigma, \Rightarrow \rangle$ and prove its completeness (a sufficient condition of the uniqueness). An element of Σ, called a state of the procedure, is the set of the current preference lists, and the reduction $\Rightarrow$ changes a state by updating the lists according to step **2.2**. To make the proof simple, we introduce an auxiliary ARS $\langle \Sigma, \rightarrow \rangle$, where the reduction $\rightarrow$ changes a state by updating the current preference list of the *single* student who is the greatest element (with respect to $\succ_j$) in the set R defined in step **2.2**. We relate the two ARSs by showing that if ARS $\langle \Sigma, \rightarrow \rangle$ is complete then ARS $\langle \Sigma, \Rightarrow \rangle$ is also complete. Then, we prove the SN and WCR properties of $\langle \Sigma, \rightarrow \rangle$ by identifying eight cases. Combining the results with Newman's lemma, we conclude that ARS $\langle \Sigma, \Rightarrow \rangle$ is complete. This means that the computational result of our procedure is unique.

2 Abstract Reduction Systems

In this section, we review the theory of ARS. An ARS is a technique that can be applied for reasoning in structures defined by equations. These reasoning abilities are important in many computer applications, including symbolic algebraic computation, automated theorem proving, program specification and verification as well as high-level programming languages. ARSs contain directed equations (called rules) which may be used as non-deterministic functional programs. *Termination* and *confluence* (i.e., *completeness*) are key properties for computing with ARSs. Termination ensures that all computations are finite and confluence ensures that the result of the computation is unique and therefore independent of the order of the rule application.

An ARS and its properties are defined as follows:

Definition 1 An ARS is a pair $\mathcal{A} = \langle A, \rightarrow \rangle$ consisting of a non-empty set A and a binary relation $\rightarrow \subseteq A \times A$. Instead of $(a, b) \in \rightarrow$ we write $a \rightarrow b$.

Definition 2

(1) The reflexive-transitive closure of $\rightarrow$ is denoted by $\twoheadrightarrow$. If $a \rightarrow b$, we say that a *reduces* or *rewrites* to b.

(2) The transitive closure of $\rightarrow$ is denoted by $\rightarrow^+$.

We write $a \leftarrow b$ if $b \rightarrow a$, likewise for $a \twoheadleftarrow b$.

Definition 3

(1) An element $a \in A$ is a *normal form* if there is no element $b \in A$ with $a \rightarrow b$. An element $a \in A$ *has a normal form* if $a \twoheadrightarrow b$ for some normal form b. The set of normal forms of $\mathcal{A}$ is denoted by $\mathrm{NF}(\mathcal{A})$ or $\mathrm{NF}(\rightarrow)$.

(2) The ARS $\mathcal{A}$ is *strongly normalizing* (SN) if there are no infinite sequences $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow \cdots$ of elements of A. In other words, every reduction sequence eventually ends in a normal form.

Definition 4

(1) An ARS $\mathcal{A} = \langle A, \rightarrow \rangle$ is *confluent* or has the *Church-Rosser property* (CR) if for all elements $a, b, c \in A$ with $b \twoheadleftarrow a \twoheadrightarrow c$, there exists some element $d \in A$ with $b \twoheadrightarrow d \twoheadleftarrow c$.

(2) An ARS $\mathcal{A} = \langle A, \rightarrow \rangle$ is *locally confluent* or has the *weakly Church-Rosser property* (WCR) if for all elements $a, b, c \in A$ with $b \leftarrow a \rightarrow c$, there exists some element $d \in A$ with $b \twoheadrightarrow d \twoheadleftarrow c$.

(3) An ARS which is SN and CR is called *complete*.

Definition 5 An ARS $\mathcal{A} = \langle A, \rightarrow \rangle$ has the *unique normal form property with respect to reduction* ($\mathrm{UN}^{\rightarrow}$) if no element of $\mathcal{A}$ reduces to different normal forms, i.e., if for all elements $a, b, c \in A$ with $b \twoheadleftarrow a \twoheadrightarrow c$ and $b, c \in \mathrm{NF}(\rightarrow)$ we have $b = c$.

Our purpose is to show $\mathrm{UN}^{\rightarrow}$ of our algorithm modelled as ARS. The following proposition shows that $\mathrm{UN}^{\rightarrow}$ is implied by CR[1].

Proposition 6 Every confluent ARS has the $\mathrm{UN}^{\rightarrow}$ property.

The following proposition shows that SN and WCR imply CR.

Proposition 7 (Newman's lemma) Every strongly normalizing and locally confluent ARS is confluent.

3 Completeness of Laboratory Assignment Algorithm

In this section, we prove the completeness of the laboratory assignment algorithm.

3.1 Formalization as ARS

Let $S = \{S_1, \dots, S_m\}$ be a set of students, and $L = \{L_1, \dots, L_n\}$ be a set of laboratories. We denote by $\succ_j$ $(1 \leq j \leq n)$ the orderings on S, where $S_a \succ_j S_b$ means that the student S_a has lower priority than the student S_b with respect to assignment to the laboratory j. We suppose that $\succ_j$ is a *strict total ordering*, that is, $\succ_j$ satisfies the following conditions for all students $S_a, S_b, S_c \in S$:

- $S_a \not\succ_j S_a$ (irreflexivity).
- If $S_a \succ_j S_b$ and $S_b \succ_j S_c$ then $S_a \succ_j S_c$ (transitivity).
- If $S_a \neq S_b$ then $S_a \succ_j S_b$ or $S_b \succ_j S_a$ (comparability).

We denote by $\langle s, (\ell_1, \dots, \ell_k)\rangle$ the list of wishes of the student s. It means that the j-th wish of the student $s \in S$ is currently $\ell_j \in L$ $(j = 1, 2, \dots, k)$. The index k is called the length of this list. At the start of the algorithm, we have $k = n$ and $(\ell_1, \dots, \ell_n)$ is a *permutation* of $(L_1, \dots, L_n)$. The set of the current lists of the wishes of all the students is called a state (of the laboratory assignment algorithm). The set of all states is called the state space, and denoted by Σ. Therefore, for all states $\sigma \in \Sigma$ and for all students $s \in S$ we have

- $\exists u, \langle s, u\rangle \in \sigma$. (That is, σ includes the list of the wishes of student s.)
- $\langle s, u\rangle \in \sigma$ and $\langle s, v\rangle \in \sigma$ imply $u = v$. (That is, σ does not include two or more lists of the wishes of student s.)

When $k > 0$ and $t = \langle s, (\ell_1, \dots, \ell_k)\rangle$, we define the operation $(\cdot)'$ by $t' = \langle s, (\ell_2, \dots, \ell_k)\rangle$. In words, removes the operation the current first wish from the list of the wishes. Moreover, we define $t'' = (t')' = \langle s, (\ell_3, \dots, \ell_k)\rangle$.

Proposition 8 In every step of the algorithm, the lists of wishes are not empty.

Proof. Suppose that the list of wishes of student s is empty $\langle s, (\ )\rangle \in \sigma$ in the state σ. This student must have been assigned to every laboratory once and, in step **2.2**, removed from there. This means that every laboratory exceeded its seating capacity before the computation reaches the state σ. Once the number of temporarily assigned students exceeded the seating capacity, it would never become less than the capacity afterwards. Therefore, in the state σ the number of temporarily assigned students of any laboratory is greater than or equal to its seating capacity. Since the number of all the students is equal to the sum of the seating capacities, the number of temporary assigned students of every laboratory must be exactly equal to its seating capacity. This means that the assignment of all students is dicided here. This contradicts that there is no assignment of the student s. $\square$

We model the laboratory assignment algorithm as ARS $\langle \Sigma, \Rightarrow \rangle$. The reduction $\Rightarrow$ changes a state according to the operation in step **2.2**. In other words, $\sigma_1 \Rightarrow \sigma_2$ $(\sigma_1, \sigma_2 \in \Sigma)$ means that σ_2 is the state obtained from σ_1 by removing the first wish from the lists of the selected students of an arbitrarily selected laboratory L_j with exceeding students. To make the proof simple, we introduce an auxiliary ARS $\langle \Sigma, \rightarrow \rangle$, where the reduction $\rightarrow$ changes a state by updating the list of wishes of a *single* student chosen from those exceeding the seating capacity. Therefore, $\sigma_1 \rightarrow \sigma_2$ $(\sigma_1, \sigma_2 \in \Sigma)$ means that the state obtained by removing the first wish of the selected student in state σ_1 is σ_2. The selection is decided according to the ordering $\succ_j$. We often write $\sigma_1 \Rightarrow_j \sigma_2$ or $\sigma_1 \rightarrow_j \sigma_2$ to specify the selected laboratory L_j explicitly.

Proposition 9

(1) If $\sigma \Rightarrow \sigma'$ then $\sigma \rightarrow^+ \sigma'$.

(2) If $\sigma \in \mathrm{NF}(\Rightarrow)$ then $\sigma \in \mathrm{NF}(\rightarrow)$.

Proof.

(1) Suppose that $\sigma \Rightarrow_j \sigma'$, i.e., the selected laboratory is L_j. If the exceeding capacity number is k $(k > 0)$, we can get k step reduction sequence $\sigma \rightarrow_j \cdots \rightarrow_j \sigma'$ by selecting L_j k times consecutively in the ARS $\langle \Sigma, \rightarrow \rangle$. Thus $\sigma \rightarrow^+ \sigma'$.

(2) If $\sigma \in \mathrm{NF}(\Rightarrow)$ then no laboratory has the assignment that exceeds its seating capacity. Thus $\sigma \in \mathrm{NF}(\rightarrow)$. □

Proposition 10 If ARS $\langle \Sigma, \rightarrow \rangle$ is complete, the ARS $\langle \Sigma, \Rightarrow \rangle$ is also complete.

Proof. Let $\rightarrow$ be complete (SN and CR). If $\Rightarrow$ is not SN, there is an infinite sequence $\sigma_1 \Rightarrow \sigma_2 \Rightarrow \cdots$, so we have an infinite sequence $\sigma_1 \rightarrow^+ \sigma_2 \rightarrow^+ \cdots$ by proposition 9 (1). This contradicts to the SN of $\rightarrow$. Therefore $\Rightarrow$ is SN.

Suppose that $\sigma_1 \Leftarrow\!\!\Leftarrow \sigma \Rightarrow\!\!\Rightarrow \sigma_2$ to show that $\Rightarrow$ is CR. ($\Rightarrow\!\!\Rightarrow$ is the reflexive-transitive closure of $\Rightarrow$.) Because $\Rightarrow$ is SN, there exist normal forms $\sigma_1', \sigma_2' \in \mathrm{NF}(\Rightarrow)$ of σ_1, σ_2, respectively. Therefore,

$$\mathrm{NF}(\Rightarrow) \ni \sigma_1' \Leftarrow\!\!\Leftarrow \sigma_1 \Leftarrow\!\!\Leftarrow \sigma \Rightarrow\!\!\Rightarrow \sigma_2 \Rightarrow\!\!\Rightarrow \sigma_2' \in \mathrm{NF}(\Rightarrow).$$

Proposition 9 shows that σ_1' and σ_2' are also normal forms of σ with respect to $\rightarrow$. Thus,

$$\mathrm{NF}(\rightarrow) \ni \sigma_1' \leftarrow^+ \sigma \rightarrow^+ \sigma_2' \in \mathrm{NF}(\rightarrow).$$

Because $\rightarrow$ has $\mathrm{UN}^{\rightarrow}$ (by proposition 6), it follows that $\sigma_1' = \sigma_2'$. Therefore $\sigma_1 \Rightarrow\!\!\Rightarrow \sigma_1' = \sigma_2' \Leftarrow\!\!\Leftarrow \sigma_2$. This means that $\Rightarrow$ is CR. □

We will show the completeness of the ARS $\langle \Sigma, \rightarrow \rangle$ in the following subsection.

3.2 Termination of $\rightarrow$

Proposition 11 ARS $\langle \Sigma, \rightarrow \rangle$ satisfies SN.

Proof. When $\sigma \rightarrow \sigma'$, the sum of the length of all lists of wishes of σ' is smaller than that of σ by one. Length of lists is non-negative. Thus there is no infinite reduction sequence $\sigma_1 \rightarrow \sigma_2 \rightarrow \cdots$. □

3.3 Confluence of $\rightarrow$

We have seen that $\langle \Sigma, \rightarrow \rangle$ is SN. By Newman's lemma, to show CR of $\langle \Sigma, \rightarrow \rangle$, it is enough to show WCR of $\langle \Sigma, \rightarrow \rangle$.

Proposition 12 ARS $\langle \Sigma, \rightarrow \rangle$ satisfies WCR.

Proof. We show the confluence of ARS $\langle \Sigma, \rightarrow \rangle$ by adjusting only L_i and L_j, the laboratories that were arbitrarily selected from those exceeding the seating capacity. That is, we show that if $\sigma_1 \leftarrow_i \sigma \rightarrow_j \sigma_2$ then $\sigma_1 \twoheadrightarrow_{i,j} \sigma' \twoheadleftarrow_{i,j} \sigma_2$ for some σ', where $\twoheadrightarrow_{i,j}$ is the reflexive-transitive closure of $\rightarrow_i \cup \rightarrow_j$. In the following, we suppose that $i = 1$ and $j = 2$ without loss of generality.

The set of the lists of wishes σ can be divided into the following five mutually-disjoint sets according to the positions where L_1 and L_2 appear.

$$A = \{t \in \sigma \mid t = \langle \ldots, (L_1, L_2, \ldots) \rangle\},$$
$$B = \{t \in \sigma \mid t = \langle \ldots, (L_1, \ldots) \rangle\} - A,$$
$$C = \{t \in \sigma \mid t = \langle \ldots, (L_2, L_1, \ldots) \rangle\},$$
$$D = \{t \in \sigma \mid t = \langle \ldots, (L_2, \ldots) \rangle\} - C,$$
$$E = \sigma - (A \cup B \cup C \cup D).$$

For example, A is the set of the lists of wishes such that the first wish is L_1 and the second wish is L_2, and B is the set of the lists of wishes such that the first wish is L_1 and the second wish is not L_2. To extend the ordering $\succ_j$ on the students to the ordering on the lists of wishes, we define $t_1 \succ_j t_2$ if and only if $s_1 \succ_j s_2$ and t_1, t_2 are lists of wishes given by $t_1 = \langle s_1, \cdots \rangle, t_2 = \langle s_2, \cdots \rangle$. Note that $t_1 \succ_j t_2$ implies $t_1' \succ_j t_2$ and $t_1 \succ_j t_2'$, i.e., the priority is not changed by removing the first wish.

Let

$$A = \{a_1, a_2, \cdots, a_p\},$$
$$B = \{b_1, b_2, \cdots, b_q\},$$
$$C = \{c_1, c_2, \cdots, c_r\},$$
$$D = \{d_1, d_2, \cdots, d_s\},$$
$$E = \{e_1, e_2, \cdots, e_t\},$$

where

$$a_1 \succ_1 a_2 \succ_1 \cdots \succ_1 a_p,$$
$$b_1 \succ_1 b_2 \succ_1 \cdots \succ_1 b_q,$$
$$c_1 \succ_2 c_2 \succ_2 \cdots \succ_2 c_r,$$
$$d_1 \succ_2 d_2 \succ_2 \cdots \succ_2 d_s.$$

We can identify eight cases given in Table I. Note that case II can be omitted from the discussion by considering the symmetry between $\succ_1$ and $\succ_2$, A and C, and B and D with case III. Similarly, VII can be omitted by considering symmetry with case V and VI.

TABLE I. Cases for proposition 12

	conditions
I	$a_1 \succ_1 b_1, c_1 \succ_2 d_1, c_1 \succ_1 a_1, a_1 \succ_2 c_1$
II	$a_1 \succ_1 b_1, c_1 \succ_2 d_1, c_1 \succ_1 a_1, c_1 \succ_2 a_1$
III	$a_1 \succ_1 b_1, c_1 \succ_2 d_1, a_1 \succ_1 c_1, a_1 \succ_2 c_1$
IV	$a_1 \succ_1 b_1, c_1 \succ_2 d_1, a_1 \succ_1 c_1, c_1 \succ_2 a_1$
V	$a_1 \succ_1 b_1, d_1 \succ_2 c_1, a_1 \succ_2 d_1$
VI	$a_1 \succ_1 b_1, d_1 \succ_2 c_1, d_1 \succ_2 a_1$
VII	$b_1 \succ_1 a_1, c_1 \succ_2 d_1$
VIII	$b_1 \succ_1 a_1, d_1 \succ_2 c_1$

We explain only the case I, which is the most complex case. The proof is summarized in Fig. 1. We first explain the left-hand side reduction path in the figure. In the first step, the laboratory L_1 is adjusted. Note that a_1 is the greatest element among $\{a_1, \ldots, a_p, b_1, \ldots, b_q\}$ with respect to $\succ_1$ by the general assumption and the condition $a_1 \succ_1 b_1$ from the case I. Thus a_1 has to give up the first wish. As the result, a_1 is removed from the set A, and a'_1, obtained from a_1 by removing the first element, is added to D instead. This changes the current state from σ to σ_1. In the second step (adjustment of L_2), a'_1 is removed from D and a''_1 added to E. Note that by $a_1 \succ_2 c_1, c_1 \succ_2 d_1$, the transitivity of $\succ_2$ and the definition of primes ($'$), we have $a'_1 \succ_2 c_1$ and $a'_1 \succ_2 d_1$. This changes the current state from σ_1 to σ'_1. In the third step (adjustment of L_2), c_1 is removed from C and c'_1 added to B. Note that by $c_1 \succ_2 d_1$. This changes the current state from σ'_1 to σ''_1. In the last step (adjustment of L_1), c'_1 is removed from B and c''_1 added to E. Note that by $c_1 \succ_1 a_1 \succ_1 a_2$ and $c_1 \succ_1 a_1 \succ_1 b_1$. This changes the current state from σ''_1 to σ'. Likewise for right-hand side of the figure. Therefore, $\sigma_1 \twoheadrightarrow \sigma' \twoheadleftarrow \sigma_2$. Similarly, we can verify the same result for all other cases. Thus ARS $\langle \Sigma, \rightarrow \rangle$ satisfies WCR. □

Theorem 13 ARS $\langle \Sigma, \rightarrow \rangle$ is complete.

Proof. It is obvious by proposition 11, 12 and 7. □

Theorem 14 ARS $\langle \Sigma, \Rightarrow \rangle$ is complete.

Proof. It is obvious by proposition 10 and theorem 13. □

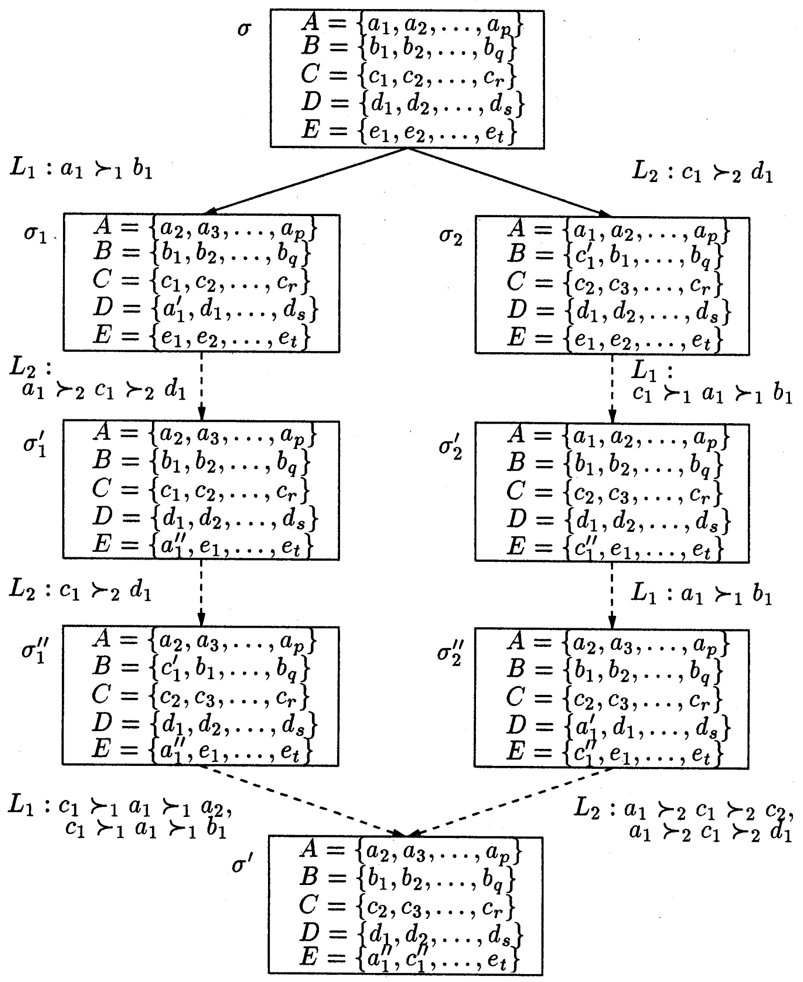
σ
$A = \{a_1, a_2, \ldots, a_p\}$ $B = \{b_1, b_2, \ldots, b_q\}$ $C = \{c_1, c_2, \ldots, c_r\}$ $D = \{d_1, d_2, \ldots, d_s\}$ $E = \{e_1, e_2, \ldots, e_t\}$
$L_1 : a_1 \succ_1 b_1$
$L_2 : c_1 \succ_2 d_1$
σ_1
$A = \{a_2, a_3, \ldots, a_p\}$ $B = \{b_1, b_2, \ldots, b_q\}$ $C = \{c_1, c_2, \ldots, c_r\}$ $D = \{a_1', d_1, \ldots, d_s\}$ $E = \{e_1, e_2, \ldots, e_t\}$
σ_2
$A = \{a_1, a_2, \ldots, a_p\}$ $B = \{c_1', b_1, \ldots, b_q\}$ $C = \{c_2, c_3, \ldots, c_r\}$ $D = \{d_1, d_2, \ldots, d_s\}$ $E = \{e_1, e_2, \ldots, e_t\}$
$L_2 :$ $a_1 \succ_2 c_1 \succ_2 d_1$
$L_1 :$ $c_1 \succ_1 a_1 \succ_1 b_1$
σ_1'
$A = \{a_2, a_3, \ldots, a_p\}$ $B = \{b_1, b_2, \ldots, b_q\}$ $C = \{c_1, c_2, \ldots, c_r\}$ $D = \{d_1, d_2, \ldots, d_s\}$ $E = \{a_1'', e_1, \ldots, e_t\}$
σ_2'
$A = \{a_1, a_2, \ldots, a_p\}$ $B = \{b_1, b_2, \ldots, b_q\}$ $C = \{c_2, c_3, \ldots, c_r\}$ $D = \{d_1, d_2, \ldots, d_s\}$ $E = \{c_1'', e_1, \ldots, e_t\}$
$L_2 : c_1 \succ_2 d_1$
$L_1 : a_1 \succ_1 b_1$
σ_1''
$A = \{a_2, a_3, \ldots, a_p\}$ $B = \{c_1', b_1, \ldots, b_q\}$ $C = \{c_2, c_3, \ldots, c_r\}$ $D = \{d_1, d_2, \ldots, d_s\}$ $E = \{a_1'', e_1, \ldots, e_t\}$
σ_2''
$A = \{a_2, a_3, \ldots, a_p\}$ $B = \{b_1, b_2, \ldots, b_q\}$ $C = \{c_2, c_3, \ldots, c_r\}$ $D = \{a_1', d_1, \ldots, d_s\}$ $E = \{c_1'', e_1, \ldots, e_t\}$
$L_1 : c_1 \succ_1 a_1 \succ_1 a_2,$ $c_1 \succ_1 a_1 \succ_1 b_1$
$L_2 : a_1 \succ_2 c_1 \succ_2 c_2,$ $a_1 \succ_2 c_1 \succ_2 d_1$
σ'
$A = \{a_2, a_3, \ldots, a_p\}$ $B = \{b_1, b_2, \ldots, b_q\}$ $C = \{c_2, c_3, \ldots, c_r\}$ $D = \{d_1, d_2, \ldots, d_s\}$ $E = \{a_1'', c_1'', \ldots, e_t\}$

Fig. 1. Case I ($c_1 \succ_1 a_1 \succ_1 b_1$; $a_1 \succ_2 c_1 \succ_2 d_1$)

4 Conclusion

We have proved the completeness (i.e., strong normalization and confluence) of our algorithm by ARSs theory. We can modify the algorithm for practical use, considering the following points:

(1) the lists of wishes of the students:

Students need not decide the entire ordering on the laboratories beforehand. Instead, each student could submit a list of three wishes for laboratories in the order of preference, and if you need, they could incrementally add more wishes.

(2) the priority on the students for each laboratory:

Laboratories need not decide the entire priority on the students beforehand. Laboratory j could decide a part of $\succ_j$, when it needs adjustment, by comparing only students that really need comparison. After this, it extends $\succ_j$ incrementally. The priority may be decided by considering factors such as the score and the strength of wishes.

(3) seating capacity:

You may decide the seating capacities after you see the submitted lists of wishes.

Reference

[1] N. Dershowitz et al., Rewrite systems, in: J. van Leeuwen ed., *Handbook of Theoretical Computer Science*, Vol. **B** (North-Holland, 1990) 243-320.