Homework 3

CS526

Joel Pfeiffer

Problem 1

$T = \{doctor, nurse, patient, healthcarerecord\}$
$TS = \{doctor, nurse, patient\}$
$TO = \{healthcarerecord\}$
$RI = \{r{:}c, w{:}c, publish{:}c\}$
$RC = \emptyset$

$link_1(doctor, nurse) = \textbf{true}$
$f_1(doctor, nurse) = \{healthcarerecord\} \times \{r{:}c, w{:}c\}$
$link_2(patient, nurse) = \textbf{true}$
$f_2(patient, nurse) = \{healthcarerecord\} \times \{publish\}$
$link_3(patient, doctor) = \textbf{true}$
$f_3(patient, nurse) = \{healthcarerecord\} \times \{publish\}$

A ticket $\mathbf{X}/r : c \in dom(\mathbf{Y})$ can be copied from $dom(\mathbf{Y})$ to $dom(\mathbf{Z})$ ifff, for some i, the following are true:

1. $\mathbf{X}/rc \in dom(\mathbf{Y})$
2. $link_i(\mathbf{Y}, \mathbf{Z})$
3. $\tau(\mathbf{X})/r{:}c \in f_i(\tau(\mathbf{Y}), \tau(\mathbf{Z}))$

Here, we can see that the subjects are TS. There are no restrictions on when record transfers are allowed, or needing a specific right to do so, other than the implicit condition that we have the right to begin with. As such all of the link functions are inherently true. We then simply use the filter function to strip down which rights are allowed, and over which entities (healthcarerecord).

Problem 2

I denote the beginning and ending time of the right with $\mathbf{X}/z_b^e$, where b represents the beginning time, and e represents the ending time for the right z over $\mathbf{X}$, where $b \leq e$ and 0 and ∞ are acceptable values.

Link Function

From definition 3-13, we know that a link predicate $link_i(\mathbf{X}, \mathbf{Y})$ is a conjunction or disjunction of the following rules r, for any $z \in RC$:

1. $\mathbf{X}/z \in dom(\mathbf{X})$
2. $\mathbf{X}/z \in dom(\mathbf{Y})$
3. $\mathbf{Y}/z \in dom(\mathbf{X})$

4. $\mathbf{Y}/z \in dom(\mathbf{Y})$
5. **true**

In order to extend this, first, I expand the function inputs to include a time input, $line\_t_i(\mathbf{X}, \mathbf{Y}, T)$. T represents the precise timing of the link predicate. I then modify the rules r to create r_t such that $r\_t = r \wedge b \leq C \leq e$. The new rules r_t, for a $z \in RC$ and given time C are:

1. $\mathbf{X}/z_b^e \in dom(\mathbf{X}) \wedge b \leq C \leq e$
2. $\mathbf{X}/z_b^e \in dom(\mathbf{Y}) \wedge b \leq C \leq e$
3. $\mathbf{Y}/z_b^e \in dom(\mathbf{X}) \wedge b \leq C \leq e$
4. $\mathbf{Y}/z_b^e \in dom(\mathbf{Y}) \wedge b \leq C \leq e$
5. $\mathbf{b} \leq T \leq \mathbf{e}$
6. **true**

Note the **b** and **e** found in rule 5. This is similar to rule 6, with the addition of being able to restrict the linkages to solely during specified times, without requiring an explicit right. Rule 6, as with the original r, simply allows linkage between two subjects.

Filter Function

A filter function is a function f_i: $TS \times TS \rightarrow 2^{T \times R}$, which has as its range the set of copyable tickets. With the time addition, our function will now take an additional parameter C, while giving out additional results, the start and end times of rights that have permissions where $b \leq C \leq e$. As such, the new filter function becomes f_t_i: $TS \times TS \times C \rightarrow 2^{T \times R \times B \times E}$.

Previously, a filter function was defined in terms of $\tilde{T}$ and $\tilde{R}$, where $\tilde{T}$ represented the *types* over which the rights $\tilde{R}$ were allowed to be transferred, per the link function $link_i$. This is represented via the notation $f_i(\tau(\mathbf{Y}), \tau(\mathbf{Z})) = \tilde{T} \times \tilde{R}$. Now, however, $\tilde{T} \times \tilde{R}$ has the additional *time* constraint. We represent this as $\left[\tilde{T} \times \tilde{R}\right]_b^e$, or the tickets for the rights $\tilde{R}$ over the types $\tilde{T}$ that begin at time b and end at time e. Our filter function can now use these values and incorporate a constraint based on time C:

$$f\_t_i\left(\tau(\mathbf{Y}), \tau(\mathbf{Z}), C\right) = \left\{ \left[\tilde{T} \times \tilde{R}\right]_b^e : b \leq C \leq e \right\}$$

We can see that this formalism allows for choosing types $\tilde{T}$ and rights $\tilde{R}$, then constraining them so they are solely allowed if the time T_i is between the start and end time of the ticket.

Copying Tickets

From pg. 68, we know that SPM only allows transference of a ticket $\mathbf{X}/r{:}c$ from $dom(\mathbf{Y})$ and $dom(\mathbf{Z})$ if three conditions are met:

1. $\mathbf{X}/rc \in dom(\mathbf{Y})$
2. $link_i(\mathbf{Y}, \mathbf{Z})$
3. $\tau(\mathbf{X})/r{:}c \in f_i(\tau(\mathbf{Y}), \tau(\mathbf{Z}))$

Now, we want to copy a right $\mathbf{X}/r{:}c$ from $dom(\mathbf{Y})$ to $dom(\mathbf{Z})$. Additionally, the time $\mathbf{Y}$ is able to have a right is not necessarily the time $\mathbf{Z}$ should be allowed to have access. For this, I denote the times that $\mathbf{Y}$ has access to a right b_y and e_y, while $\mathbf{Z}$ wants access to the right at from time b_z to time e_z. Here, I make an additional restriction - $\mathbf{Y}$ *cannot* assign a right to $\mathbf{Z}$ for times which $\mathbf{Y}$ does not have the access. If this was not enforced, then $\mathbf{Y}$ could get timed access to a right that was not intended, through using $\mathbf{Z}$.

We can now tie this with the link function and filter function definitions. We say a ticket $\mathbf{X}/r{:}c_{b_y}^{e_y}$ that $\mathbf{Y}$ holds between b_y and e_y can be copied from $dom(\mathbf{Y})$ to $dom(\mathbf{Z})$ where it can be accessed by $\mathbf{Z}$ between times b_z and e_z (represented by $\mathbf{X}/r{:}c_{b_z}^{e_z}$) during a time C iff, for some i, the following are true:

1. $\mathbf{X}/rc_{b_y}^{e_y} \in dom(\mathbf{Y}) \wedge b_y \leq b_z \wedge e_y \geq e_z$
2. $link\_t_i(\mathbf{Y}, \mathbf{Z}, C)$
3. $\tau(\mathbf{X})/r{:}c_{b_y}^{e_y} \in f_i(\tau(\mathbf{Y}), \tau(\mathbf{Z}), C)$

Here, we have stated that a) we have the ticket, and the time we are transferring for is during the time we are allowed the ticket, b) we are allowed to link from $\mathbf{Y}$ to $\mathbf{Z}$ at time C, and c) the right we are trying to copy is allowed to be copied at this specific instance in time. As such, we have effectively constrained SPM to only allow the transfer of tickets during allowed times.

Creation

Now, we need to be able to allow people to create items. In the original SPM, we simply declared:

$$cc(a, b)$$

and that implicitly allowed subjects of type a to create an entity of type b. With the time-based model, we might want to only allow the creation of a new entity at a specific *time*. To deal with this, I use the approach the linkage uses, by defining rules which can be combined via conjunction or disjunction. So, for a function $cc(a, b, C)$, where C is the time, we only have two rules:

1. $\mathbf{b} \leq C \leq \mathbf{e}$
2. **true**

$\mathbf{b}$ and $\mathbf{e}$ are times which can be set. Note that this allows us to join multiple times together, such as $\mathbf{b}_1 \leq C \leq \mathbf{e}_1 \vee \mathbf{b}_2 \leq C \leq \mathbf{e}_2$, meaning we can create during one time, or another.

Lastly, we have the rules for what permissions are added to both the parent and child when the child is created. In the non-temporal SPM, we would list a set of tickets that each would get via

$$cr_p(a, b) = \left\{\tilde{R}a\right\}$$

$$cr_c(a, b) = \left\{\tilde{R}b\right\}$$

Now, however, we need to specify the times which rules are allowed. This results in the addition of the usual $\mathbf{b}$ and $\mathbf{e}$, which need to be specified by the person who writes the rules. The syntax for this would be:

$$cr_p(a, b, C) = \left\{\tilde{R}a_{\mathbf{b}}^{\mathbf{e}}\right\}$$

$$cr_c(a, b, C) = \left\{\tilde{R}b_{\mathbf{b}}^{\mathbf{e}}\right\}$$

This specifies, for a particular time C, what rules would we like to apply, meaning we have the flexibility to have different rule sets created at different times.

Problem 3

For this problem, I choose to use the HRU model. This involves the use of people being the subjects, and the individual problems, group problems, individual results and group results all being objects. In order to distinguish between the types of objects, the objects are given a right corresponding to its type. For instance if we take the Individual Problem 1, this corresponds to the column IP_1 in our ACM, and the right ip_1 is then present in every row under that column for the initial matrix M_0. In addition to the Individual Problem type,

	I	S_1	S_2	...	S_n	IP_1	...	IP_5	GP_1	GP_2
I		t	t	...	t	ip_1, rw	...	ip_5, rw	gp_1, rw	gp_2, rw
S_1	t	$ir_1, ir_2, ir_3, ir_4, ir_5, gr_1, gr_2$				ip_1, r	...	ip_5, r	gp_1, r	gp_2, r
S_2	t		$ir_1, ir_2, ir_3, ir_4, ir_5, gr_1, gr_2$			ip_1, r	...	ip_5, r	gp_1, r	gp_2, r
$\vdots$	$\vdots$			$\ddots$		$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
S_n	t				$ir_1, ir_2, ir_3, ir_4, ir_5, gr_1, gr_2$	ip_1, r	...	ip_5, r	gp_1, r	gp_2, r

Table 1: Our initial ACM for determining a test with 5 individual problems and 2 group problems. Note that each student is given ir and gr for each problem initially.

we can also have Individual Response (for the student's answer to individual problems), Group Problem and Group Response, with the corresponding rights ir, gp, and gr (with a subscript indicating which problem it is associated with).

In addition to the rights used to explicitly define available problems and solutions, we also have the rights t for talk, r for read and w for write. I allow the students to converse with the professor, but not with each other. Additionally, the professor is allowed to read and write each of the problems, but students are only allowed to read the problems. This is so we don't have a student modifying what the question is. In order to get the students' responses to the questions, we use the ir and gr rights, creating new objects (which contain the students answers). Overall, our set of rights is:

$$Rights = \{t, r, w, ip_1, ip_2, ip_3, ip_4, ip_5, gp_1, gp_2, ir_1, ir_2, ir_3, ir_4, ir_5, gr_1, gr_2\}$$

We can now look at Table 1, which is our initial access control matrix. We see that the the instructor is allowed to speak with any student, as they are allowed to speak with the instructor. Additionally, the instructor and read and write each of the problems, while the students are only given read access. Finally, we have given an 'indicator' right to each problem for each student, so we can be aware of which problem they are doing.

Next, we need to allow students to answer the *individual* problems. For this, we create 5 similar functions, which ensure the student is a) answering the correct problem and b) has not answered the problem previously. We do not want a student to be able to submit many answers to a single problem.

// Answering individual problem 1, we take the student, instructor, problem, and response
Command *Answer_Ind_Problem_One*($subject_1, subject_2, object_1, object_2$)
if ir_1 **in** $a[subject_1, subject_1]$ *// Have we already answered problem 1?*
and t **in** $a[subject_1, subject_2]$ *// Is the other person the instructor?*
and ip_1 **in** $a[subject_1, object_1]$ *// Are we answering the right problem?*
and r **in** $a[subject_1, object_1]$ *// Can we read the test?*
then
    create object $object_2$; *// Create the answer*
    enter r **into** $a[subject_1, object_2]$; *// Give student permissions to the answer*
    enter w **into** $a[subject_1, object_2]$;
    enter r **into** $a[subject_2, object_2]$; *// Give instructor permission to read the answer*
    enter ir_1 **into** $a[subject_1, object_2]$; *// 'Label' the answer*
    enter ir_1 **into** $a[subject_2, object_2]$; *// 'Label' the answer*
    delete ir_1 **from** $a[subject_1, subject_1]$; *// Remove student's permission to create a new answer*
end

// Answering individual problem 2, we take the student, instructor, problem, and response
Command *Answer_Ind_Problem_Two*($subject_1, subject_2, object_1, object_2$)

if ir_2 **in** $a[subject_1, subject_1]$ *// Have we already answered problem 2?*
and t **in** $a[subject_1, subject_2]$ *// Is the other person the instructor?*
and ip_2 **in** $a[subject_1, object_1]$ *// Are we answering the right problem?*
and r **in** $a[subject_1, object_1]$ *// Can we read the test?*
then
    create object $object_2$; *// Create the answer*
    enter r **into** $a[subject_1, object_2]$; *// Give student permissions to the answer*
    enter w **into** $a[subject_1, object_2]$;
    enter r **into** $a[subject_2, object_2]$; *// Give instructor permission to read the answer*
    enter ir_2 **into** $a[subject_1, object_2]$; *// 'Label' the answer*
    enter ir_2 **into** $a[subject_2, object_2]$; *// 'Label' the answer*
    delete ir_2 **from** $a[subject_1, subject_1]$; *// Remove student's permission to create a new answer*
end

// Answering individual problem 3, we take the student, instructor, problem, and response
Command $Answer\_Ind\_Problem\_Three(subject_1, subject_2, object_1, object_2)$
if ir_3 **in** $a[subject_1, subject_1]$ *// Have we already answered problem 3?*
and t **in** $a[subject_1, subject_2]$ *// Is the other person the instructor?*
and ip_3 **in** $a[subject_1, object_1]$ *// Are we answering the right problem?*
and r **in** $a[subject_1, object_1]$ *// Can we read the test?*
then
    create object $object_2$; *// Create the answer*
    enter r **into** $a[subject_1, object_2]$; *// Give student permissions to the answer*
    enter w **into** $a[subject_1, object_2]$;
    enter r **into** $a[subject_2, object_2]$; *// Give instructor permission to read the answer*
    enter ir_3 **into** $a[subject_1, object_2]$; *// 'Label' the answer*
    enter ir_3 **into** $a[subject_2, object_2]$; *// 'Label' the answer*
    delete ir_3 **from** $a[subject_1, subject_1]$; *// Remove student's permission to create a new answer*
end

// Answering individual problem 4, we take the student, instructor, problem, and response
Command $Answer\_Ind\_Problem\_Four(subject_1, subject_2, object_1, object_2)$
if ir_4 **in** $a[subject_1, subject_1]$ *// Have we already answered problem 4?*
and t **in** $a[subject_1, subject_2]$ *// Is the other person the instructor?*
and ip_4 **in** $a[subject_1, object_1]$ *// Are we answering the right problem?*
and r **in** $a[subject_1, object_1]$ *// Can we read the test?*
then
    create object $object_2$; *// Create the answer*
    enter r **into** $a[subject_1, object_2]$; *// Give student permissions to the answer*
    enter w **into** $a[subject_1, object_2]$;
    enter r **into** $a[subject_2, object_2]$; *// Give instructor permission to read the answer*
    enter ir_4 **into** $a[subject_1, object_2]$; *// 'Label' the answer*
    enter ir_4 **into** $a[subject_2, object_2]$; *// 'Label' the answer*
    delete ir_4 **from** $a[subject_1, subject_1]$; *// Remove student's permission to create a new answer*
end

// Answering individual problem 5, we take the student, instructor, problem, and response
Command $Answer\_Ind\_Problem\_Four(subject_1, subject_2, object_1, object_2)$
if ir_5 **in** $a[subject_1, subject_1]$ *// Have we already answered problem 5?*
and t **in** $a[subject_1, subject_2]$ *// Is the other person the instructor?*
and ip_5 **in** $a[subject_1, object_1]$ *// Are we answering the right problem?*

and r **in** $a[subject_1, object_1]$ *// Can we read the test?*
then
create object $object_2$; *// Create the answer*
enter r **into** $a[subject_1, object_2]$; *// Give student permissions to the answer*
enter w **into** $a[subject_1, object_2]$;
enter r **into** $a[subject_2, object_2]$; *// Give instructor permission to read the answer*
enter ir_5 **into** $a[subject_1, object_2]$; *// 'Label' the answer*
enter ir_5 **into** $a[subject_2, object_2]$; *// 'Label' the answer*
delete ir_5 **from** $a[subject_1, subject_1]$; *// Remove student's permission to create a new answer*
end

Walking through each of the individual functions, we can see that four comparisons happen. First, we check whether or not we have already answered the problem. Second, we check whether the other subject passed in was the instructor, leveraging the fact that only the instructor can talk to the student. Next, we check that we are answering the correct problem (which won't affect the problem), and finally, we check whether or not we can actually read the problem (maybe the test has ended).

After these checks are done, the creation is straightforward, we just create the object, giving the student r/w permissions, as well as giving the instructor read permissions (for grading). We then remove the student's ability to create a new answer. This means that should we ever call this function again with the same parameters, it will fail, because the student has already answered the question. Note that the student is still free to modify the actual answer, they just cannot create a *new* one.

For the next section, we use similar logic to extend how many students are allowed to edit a problem.

// Answering group problem 1, we take the student, instructor, problem, and response
Command $Answer\_Grp\_Problem\_One(subject_1, subject_2, subject_3, object_1, object_2)$
if gr_1 **in** $a[subject_1, subject_1]$ *// Has student 1 already answered group problem*
and gr_1 **in** $a[subject_2, subject_2]$ *// Has student 2 already answered group problem*
and t **in** $a[subject_1, subject_3]$ *// Is the other person the instructor*
and t **in** $a[subject_2, subject_3]$ *// Is the other person the instructor*
and gp_1 **in** $a[subject_1, object_1]$ *// Are we answering the right problem*
and gp_1 **in** $a[subject_2, object_1]$ *// Are we answering the right problem*
and r **in** $a[subject_1, object_1]$ *// Can student 1 read the test?*
and r **in** $a[subject_2, object_1]$ *// Can student 2 read the test?*
then
create object $object_2$; *// Create the answer*
enter r **into** $a[subject_1, object_2]$; *// Give student 1 permissions to the answer*
enter w **into** $a[subject_1, object_2]$;
enter r **into** $a[subject_2, object_2]$; *// Give student 2 permissions to the answer*
enter w **into** $a[subject_2, object_2]$;
enter r **into** $a[subject_2, object_2]$; *// Give instructor permission to read the answer*
enter gr_1 **into** $a[subject_1, object_2]$; *// 'Label' the answer*
enter gr_1 **into** $a[subject_2, object_2]$; *// 'Label' the answer*
enter gr_1 **into** $a[subject_3, object_2]$; *// 'Label' the answer*
delete gr_1 **from** $a[subject_1, subject_1]$; *// Remove student 1's permission to create a new group answer*
delete gr_2 **from** $a[subject_1, subject_1]$; *// Remove student 1's permission to create a new answer (for group problem 2)*
delete gr_1 **from** $a[subject_2, subject_2]$; *// Remove student 2's permission to create a new group answer*
delete gr_2 **from** $a[subject_2, subject_2]$; *// Remove student 2's permission to create a new answer (for group problem 2)*
end

// Answering group problem 2, we take the student, instructor, problem, and response

	I	S_1	S_2	...	S_n	IP_1	...	GP_2	$IR_1^{S_1}$	$GR_2^{S_1,S_2}$
I		t	t	...	t	ip_1, r, w	...	ip_2, r, w	ir_1, r	gr_2, r
S_1	t	$ir_2, ir_3,$ $ir_4, ir_5,$				ip_1, r	...	gp_2, r	ir_1, r, w	gr_2, r, w
S_2	t		$ir_1, ir_2, ir_3,$ $ir_4, ir_5,$			ip_1, r	...	gp_2, r		gr_2, r, w
$\vdots$	$\vdots$			$\ddots$		$\vdots$	$\ddots$	$\vdots$		
S_n	t				$ir_1, ir_2, ir_3,$ $ir_4, ir_5,$ gr_1, gr_2	ip_1, r	...	gp_2, r		

Table 2: Our ACM after student 1 has created an answer for problem 1, and after student 1 and student 2 together have created an answer for group problem 2.

Command *Answer_Grp_Problem_Two*($subject_1, subject_2, subject_3, object_1, object_2$)
if gr_2 **in** $a[subject_1, subject_1]$ *// Has student 1 already answered group problem*
and gr_2 **in** $a[subject_2, subject_2]$ *// Has student 2 already answered group problem*
and t **in** $a[subject_1, subject_3]$ *// Is the other person the instructor*
and t **in** $a[subject_2, subject_3]$ *// Is the other person the instructor*
and gp_2 **in** $a[subject_1, object_1]$ *// Are we answering the right problem*
and gp_2 **in** $a[subject_2, object_1]$ *// Are we answering the right problem*
and r **in** $a[subject_1, object_1]$ *// Can student 1 read the test?*
and r **in** $a[subject_2, object_1]$ *// Can student 2 read the test?*
then
create object $object_2$; *// Create the answer*
enter r **into** $a[subject_1, object_2]$; *// Give student 1 permissions to the answer*
enter w **into** $a[subject_1, object_2]$;
enter r **into** $a[subject_2, object_2]$; *// Give student 2 permissions to the answer*
enter w **into** $a[subject_2, object_2]$;
enter r **into** $a[subject_2, object_2]$; *// Give instructor permission to read the answer*
enter gr_2 **into** $a[subject_1, object_2]$; *// 'Label' the answer*
enter gr_2 **into** $a[subject_2, object_2]$; *// 'Label' the answer*
enter gr_2 **into** $a[subject_3, object_2]$; *// 'Label' the answer*
delete gr_1 **from** $a[subject_1, subject_1]$; *// Remove student 1's permission to create a new answer (for group problem 1)*
delete gr_2 **from** $a[subject_1, subject_1]$; *// Remove student 1's permission to create a new answer gr2*
delete gr_1 **from** $a[subject_2, subject_2]$; *// Remove student 2's permission to create a new answer (for group problem 1)*
delete gr_2 **from** $a[subject_2, subject_2]$; *// Remove student 2's permission to create a new answer gr2*
end

Looking at these group problems, we can see that (again) there is a list of requirements. Both students must have not previously answered a group problem, both students must have the instructor available for communication, both students must be answering the correct problem, and both students must be able to read the test. After these requirements are met, the answer is created, with both students being able to r/w the answer, while the instructor can read it. Additionally, the students are not allowed to answer any more group problems after this (either 1 or 2). So, if they are ever called to pair with another student, it will be denied.

We can see that the usage of these rules, along with the initial starting matrix, allows each student to create one result for each individual problem. Creation of an answer prevents removes the right that allowed the creation, so only one will ever be created. r/w rights on the answer are given to the student, while

the instructor gets read rights. Additionally, for the group problem, a student can participate in one group problem, which results in permissions being taken away from both, disallowing them to work with each another student. We note that at no point does t change, so we know that no one can cheat by speaking with another person.

In Table 2, we see the effects after student 1 creates an assignment response for problem 1, then teams with student 2 to answer group problem 2. We see that student 1 has lost ir_1 in $a[S_1, S_1]$, while both have lost gr_2. Two additional objects have been created, which is their group answer.

I chose the HRU model for its flexibility in removing a right. Specifically, I was able to define a function which deleted the right as part of the function. This restricted any student from creating multiple copies, and from working with multiple students. The SPM does not allow this deletion – once a right is given, it is assumed to be there forever. SPM also requires the link function to be in terms of concatenation of rights present via conjunction or disjunction, not is explicitly forbidden, so we can not add a right that restricts the actions. The Extended SPM allows for mutual creation, but HRU also allows this. Additionally, TG does not have the kind of restrictive capabilities necessary – once a grant is given, it cannot be taken away, and students can grant permission to read their problems to anyone else. While it is true that TG can have the 'overload', or instructor, grant permissions to students to share a homework (see the buffer example) this implies that the instructor chooses who is working with who, and on what project. With HRU, students answering the group problems lies in their domain, rather than being granted access.