Gruppenarbeit
Semantik-basierte Modellierung

Petar Byulbyulev (a1254014), Andrii Fedorenko (a1349252)

Universität Wien, UE Anwendungen aus semantischen Technologien
Professor: Priv.-Doz. Mag. Dr. Hans-Georg Fill

1 Konzept

Für unsere Gruppenarbeit haben wir für die Ausführung Prolog ausgewählt. Wir haben zwei verschiedene BPMN Modelle. Das Erste beschreibt ein Check-In Prozess am Flughafen und das Zweite ein Besuch beim Arzt. Aus den beiden XML Files haben wir verschiedene Fakten gesammelt. Wir haben bemerkt, dass alle Elemente in dem BPMN die gleiche Struktur haben und dass den ganzen BPMN von ein paar Elemente aufgebaut ist:

- Start Event
- Exclusive Gateway
- Task
- End Event

Im XML File kann man leicht die verschiedenen Elemente trennen. Am Anfang sind die "MODELATTRIBUTES", wo der ganze BPMN beschrieben ist (z.B. Anzahl von Elementen, Autor und Typ von Diagramm). Danach kommt jede einzelne "INSTANCE", wo man verschiedene Informationen über die Attributen von dieser Instance finden kann. Am Ende sind alle Verbindungen zwischen den Elementen beschrieben. Wir haben bemerkt, dass man viele verschiedene Fakten von den Attributen schreiben kann, aber eigentlich sind die Meisten gleich für alle Elemente und es wäre sinnlos darüber Regeln zu schreiben. Deshalb haben wir entschieden, uns eher auf den Pfad von einem zu dem anderen Objekt zu konzentrieren.

2 Ausführung

Für die Ausführung muss man folgende Schritte machen:

- cd team
- Erste BPMN Model: java -jar SaxonEE9-7-0-15J/saxon9ee.jar -xsl:transf.xsl -s:bpmn_process_1.xml -o:bpmn1.pl
- Zweite BPMN Model: java -jar SaxonEE9-7-0-15J/saxon9ee.jar -xsl:transf.xsl -s:bpmn_process_2.xml -o:bpmn2.pl

Da werden die zwei Modelle kompiliert. Mittels Transformation von xslt File (transf.xslt) als Ausgabe werden zwei Prolog Files bpmn1.pl und bpmn2.pl.

In *.pl kann man auch Kommentare mit Beispielaufrufe finden.

3 XSLT Transformation

Mittels XPATH und XSLT Ausführung Befehle haben wir BPMN xml Files zu Prolog Format transformiert. Die Hauptidee ist alle Parameters zu adoptieren. Z.B. Instanz mit dem Name "Online Check-in already completed?" wird ins "online_check-in_already_completed" transformiert.

Zeiten werden auch mittels REGEX Expressions transformiert. Z.B. 00:001:01:00:00 wird auf Jahren, Tagen, Minuten und Sekunden geteilt. Das heißt im diesem Fall:

- has_max_start_period_years(agent_transfers_baggage_to_conveyor,00).
- has_max_start_period_days(agent_transfers_baggage_to_conveyor,001).
- has_max_start_period_hours(agent_transfers_baggage_to_conveyor,01).
- has_max_start_period_minutes(agent_transfers_baggage_to_conveyor,00).
- has_max_start_period_seconds(agent_transfers_baggage_to_conveyor,00).

4 Fakten

Zu erst haben wir Fakten über die ganze BPMN Struktur geschrieben wie zum Beispiel Anzahl von Elementen im BPMN, Autor, Model, Typ und danach haben wir die Fakten für jeden einzelnen Element geschrieben:

- Typ von Element
- Position in BPMN Struktur
- Order
- Variable Scope
- Variable Typ
- Priorität
- Execution Time
- Waiting Time
- Resting Time
- Transport Time
- Max Resource Waiting Time
- Min Quota of Presence
- Max Start Period
- Start Quantity

Manche von diesen Fakten sind nur für Elemente von Typ "Exclusive Gateway", andere für Elemente von Typ "Task" (Wie zum Beispiel Execution Time, Waiting Time, Resting Time, Transport Time und so weiter). Danach haben wir Fakten für alle Verbindungen geschrieben (zum Beispiel subsequent(check-in_process_airport,online_checkin_already_completed)). So war es möglich die verschiedenen Elementen zu trennen und zu beschreiben wie sie verbunden sind und was der Pfad von ein bis zu dem anderen Element ist. Das haben wir bei den Regeln gemacht.

5 Regeln

Wir haben die folgenden Regeln:

- show_bpmn_attr(X) - Diese Regel gibt gemeinsame Information über alle BPMN Attributen, wie zum Beispiel Name von Model, Autor, Anzahl von Objekten im ganzen BPMN und Typ von Modell aus.
- show_obj_attr(A) - Diese Regel beschreibt die Hauptattributen von jedem Element, es ist nicht spezifiziert, um was für ein Element es geht. Alle verschiedene Elemente in unserer BPMN Struktur haben diese Attributen.
- path_exist(A,B) - Diese Regel überprüft, ob es ein Pfad existiert zwischen zwei verschiedenen Elementen. Wenn ein Pfad existiert, es gibt als Ergebnis "true" zurück, wenn nicht "false".
- show_path(A,B) - Diese Regel zeigt jeder Schritt von Pfad, von einem bis zu dem anderen Element. Wenn es mehr als eine Möglichkeit gibt, wählt diese Regel der Pfad auf Prinzip First-come, first-served (FCFS).
- show_total_exec_time(A,B,S_YEARS,S_DAYS,S_HOURS,S_MINUTES) - Diese Regel kalkuliert die gemeinsame "Execution Time" von Element "A" bis Element "B", diese Werte sind nur für Elemente von Typ "Task" genommen.
- show_total_waiting_time(A,B,S_YEARS,S_DAYS,S_HOURS,S_MINUTES) - Diese Regel ist genau wie die Vorige nur für Elemente von Typ "Task". Sie kalkuliert die gemeinsame "Waiting Time".
- show_total_resting_time(A,B,S_YEARS,S_DAYS,S_HOURS,S_MINUTES) - kalkuliert die gemeinsame "Resting Time" zwischen Element A und B.
- show_total_transport_time(A,B,S_YEARS,S_DAYS,S_HOURS,S_MINUTES) - kalkuliert die gemeinsame "Transport Time".
- show_total_max_resource_waiting_time(A,B,S_YEARS,S_DAYS,S_HOURS,S_MINUTES) - kalkuliert die gemeinsame "Max Resource Waiting Time".
- show_total_max_start_period(A,B,S_YEARS,S_DAYS,S_HOURS,S_MINUTES) - kalkuliert die gemeinsame "Max Start Period".
- show_exec_time_of_task(A) - Zeigt die "Executin Time" von einem spezifischen Task in Jahren, Tagen, Stunden, Minuten und Sekunden.
- show_waiting_time_of_task(A) - zeigt die "Waiting Time" von einem Task.
- show_resting_time_of_task(A) - "Resting Time" von einem Task.
- show_transport_time_of_task(A)
- show_max_resource_waiting_time_of_task(A)
- show_max_start_period_of_task(A)