Extracting Declare Constraints from Natural Language

This document describes the functionality of the Speech2RuM modeling tool. Section 1 describes the steps that users should take to specify, extend, and adapt declarative process constraints. Section 2 specifies basic constraint types supported by the tool. Section 3 describes enriched version of these types, e.g., using negation or chain constraints, whereas Section 4 describes how constraints can be augmented with data and time conditions.

1. VOICE TOOL FLOW

To enter voice control menu:

1. Enter the “MP-Declare Editor” view and select “New MP-Declare Model”.

Creating a constraint using the speech tool:

1. Press “Record”.
2. The tool will now listen for the next input sentence from the microphone. It is possible to cancel the recording mid-way by pressing the “Cancel” button.
3. Once the tool has recognised a sentence, it will display the sentence in the text box above the “Record” button, and will add the detected activities and constraint(s) to the corresponding areas.
4. If needed, it is possible to edit the activities by clicking the button to the right of each activity, the same for constraints. While editing, it is possible to change the wording of the actions, the constraint type, or add conditions to constraints (see Section 4).
5. The user can repeat Steps 1 to 4 to add additional constraints.

2. SUPPORTED BASE CONSTRAINT TYPES

The current implementation supports nine base constraint types: INIT, EXISTENCE, ABSENCE, PRECEDENCE, RESPONSE, SUCCESSION, COEXISTENCE, RESPONDED EXISTENCE, AT-MOST-ONCE. This section describes these types and the input required to express them in natural language.

INIT - Defines the activity that should happen first for a process instance.

Input req.: A sentence that contains a process-related noun, e.g., “process”, “instance”, “case”, and a verb denoting that something starts, e.g., “start”, “begin”.

Examples:

- Init(customer arrives) - “The **process starts** when the customer arrives”
- Init(receive order) - “A **case begins** when an order is received”

EXISTENCE - Defines an activity that should occur at least once per process instance.

Input req.: A sentence describing a single activity that includes an obligation, e.g., “must”, “should”, “will”.

Examples:

- Existence(create an invoice) - “An invoice must be created”
- Existence(each product pass a test) - “Each product should pass a test”

ABSENCE - Defines an activity that should never occur.

Input req.: A sentence describing a single activity including a negation.

Examples:

- Absence(fail tests) - “Tests should not be failed”
- Absence(allow dogs) - “Dogs are not allowed in the restaurant”

PRECEDENCE - A Precedence(target, activation) constraint defines that an activity (activation) can only occur if another activity (target) happened beforehand.

Input req: A sentence that invokes an order between two activities, for which the target activity is not mandatory, e.g., not indicated with “can”, “may”, “could”.

Examples:

- Precedence(create a claim, approve a claim) - “A claim must be created before it is approved”
- Precedence(create a claim, approve a claim) - “If a claim is approved, then it must have been created first”

RESPONSE - A Response(activation, target) constraint defines that if some activity happens (activation), another activity (target) must be executed later.

Input req.: A sentence that invokes an order between two activities, for which the target activity is mandatory, e.g., indicated with “must”, “should”, “will”, and which indicates order, e.g. with “then”, "after", "later".

Examples:

- Response(ship order, send invoice) - “When an order is shipped then an invoice must be sent.”
- Response(close building, staff leaves) - “The staff must leave after the building is closed.”

SUCCESSION - A Succession(activation, target) constraint defines that an activity (activation) can only happen if and only if another activity (target) happens later.

As SUCCESSION constraints are syntactic sugar to express the conjunction of a precedence and a response, we have proposed a way to create such constraints by creating a base RESPONSE/PRECEDENCE constraint and changing the constraint type.

CO-EXISTENCE - A Co-Existence(activation, target) constraint defines that two activities should always be executed together for a process instance.

Input req.: A sentence that describes two actions that should co-occur.

Examples:

- Co-Existence(order shipping, invoice payment) - “Order shipping and invoice payment should occur together”

RESPONDED EXISTENCE - A RespondedExistence(activation, target) constraint defines that if an activity (activation) occurs, then another activity (target) must occur.

Input req.: A sentence that specifies two activities, for which the target activity is mandatory, e.g., indicated with “must”, “should”, “will”, and which does not indicate order

Examples:

- Responded Existence(produce product, test the product) - “If a product is produced, the product must be tested.”
- Responded Existence(identify an incident, log the details) - “When an incident is identified, the details must be logged”

AT-MOST-ONCE / ABSENCE2 - defines that a certain activity should be executed at most once per process instance. Displayed in RuM as “Absence2”.

Input req.: A sentence describing a single activity that indicates an execution limit.

Examples:

- AtMostOnce(invoice paid) - “An invoice should be paid at most once”
- AtMostOnce(invoice paid) - “An invoice should not be paid more than once”

3. FURTHER CONSTRAINT TYPES

NOT-SUCCESSION/NOT COEXISTENCE - define that two activities cannot succeed each other (Not Succession) or cannot occur in the same process instance.

Req: A sentence describing two activities where the **activation action** must be negative (include “not”)

Examples:

- Not Succession(an error appear, we react) - “if an error appears, we **will not** react”
- Not Succession(clear the results, notify the employee) - “after the results are cleared, the employee is **not** notified”
- Not Coexistence(accept application, reject application) - “if an application is accepted, it cannot be rejected”

CHAIN-Constraints - use the logic of the original constraint, but the activation and target action must take place immediately after each other (no other action can happen in between)

Req: RESPONSE/PRECEDENCE constraint, where the **target action** is linked to an “immediate” modal. NOT and CHAIN constraints can be used together.

Immediate modals: “immediately”, “instantly”, “directly”, “promptly”

Examples:

- Chain Precedence(receive the notification, display the results) - “after the notification is received, the results are displayed **immediately**”
- Not Chain Response(an error appear, we react) - “if an error appears, we will not react **instantly**”

ALTERNATE-Constraints - use the logic of the original constraint, but the activation/target action cannot reoccur before the constraint is satisfied

As ALTERNATE constraints don’t usually appear in natural language (e.g. “if action A happens, then it does not happen again before action B happens”), we have proposed a way to create such constraints by creating a base RESPONSE/PRECEDENCE constraint and changing the constraint type.

4. CONDITION TYPES

Conditions can be added to a constraint after the constraint type and actions have been set.

ACTIVATION CONDITION - Specifies a constraint on the activation activity.

Req: numerical condition: field is (not) (greater, lower than (or equal to)) value; categorical condition: field (not) in list of values

Examples:

- “A.value > 4” - “value is greater than four”
- “A.type not in (basic, medium, large)” - “type not in basic, medium, large”

CORRELATION CONDITION - specifies a constraint that considers a field of both the activation and target activity.

Req: same/different field

Examples:

- “same price”
- “different resource”

TIME CONDITION - specifies a constraint that considers the time elapsed between the activation and target action

Req: between X and Y seconds/minutes/hours/days

Examples:

- “5,13,m” - “between 5 and 13 minutes”
- “0,2,d” - “between 0 and 2 days”