1 SPDL

SPDL [1] is a service modeling language unified three flows (work flow, data flow and value flow) from three perspectives (control-flow perspective, data perspective and resource perspective). The important elements are defined in this section by Type Theory. Some notations used in this section can be found in Tab. **??**. The paragraph started with **Example** means the example of corresponding definition in *Youku*.

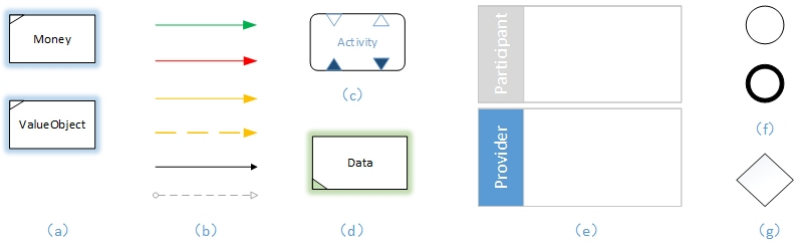

Fig. 1. The notations of SPDL diagram.

1.1 Resource

There are two kinds of resources: **money** and **value object**. The goods, visual item and intangible resource are value objects. The brand is a special value object which is used in licensing fee pattern (see section **??**). The formalization of resource in Backus-Naur-Form (BNF) is present in Equation 1

$$Resource := Money|(\lambda id_r : nat.ValueObject) \tag{1}$$

The function $\lambda id_r : nat.ValueObject[id_r]$, constructor of the value object with the parameter id_r , helps to define each value object. The *nat* is short for nature number.

Example: The video, can be defined in Eq. 2, is an important kind of value object (1431 is the global id). The global id is a nature number used in many examples follows. The notations in Fig. 1 (a) are **money** and **value object** (from top to bottom).

$$Video := (\lambda id_r : nat.ValueObject) \quad 1431 \tag{2}$$

There are four basic operations on the resource: create, use, exclusively use and consume.

- **Create**: generate a resource from an activity.
- **Use**: use a resource in an activity.

– **Exclusively use**: use a resource in an activity, but forbid using from other actors at the same time.
– **Consume**: use up a resource in an activity.

The operations (create, use, exclusively use and consume) on resource belong to the type of a *Cartesian Product*:

$$f_{Op_r} := \nu_{cr}[\![Create_r]\!]|\nu_{ur}[\![Use_r]\!]|\nu_{er}[\![ExclusiveUse_r]\!]|\nu_{or}[\![Consume_r]\!] : \prod r : Resource.Op_r(r) \tag{3}$$

In Eq. 3, ν presents an operation, the r in subscript means the resource (differing from data below).

Example: Watching a video can be expressed as follows:

$$WatchVideo := \nu_{ur} \quad Video \tag{4}$$

1.2 Data

Data is the attribute of the entity in the service. The formalization of data in BNF is present in Eq. 5

$$Data := \lambda id_d : nat.\mu_d[\![DataConstructor]\!] \tag{5}$$

In the Eq. 5, μ is used to declare a constructor, the subscript, d, means that its construct a data type. The notation in Fig. 1 (d) is data. Similar to the operation in resource, there are four operations on data. The type of them are also *Cartesian Product* type.

$$f_{Op_d} := \nu_{cd}[\![Create_d]\!]|\nu_{ud}[\![Use_d]\!]|\nu_{ed}[\![ExclusiveUse_d]\!]|\nu_{od}[\![Consume_d]\!] : \prod d : Data.Op_d(d) \tag{6}$$

Example: The identification of customer is a important data. A customer can be either a normal one or a VIP. The global id of VIP is 2432 and Normal 2433.

$$VIP := \mu_d \quad 2432 \qquad\qquad Normal := \quad \mu_d \quad 2433 \tag{7}$$

1.3 Actor

There are two kinds of actor: the **service provider** and the **participant**. The actor is formalized as follows:

$$Actor := Provider|(\lambda id_a : nat.Participant) \tag{8}$$

A participant can be constructed by β reduction.

$$(\lambda id_a : nat.Participant)id = Participant[id_a/id] \tag{9}$$

The actor is present in Fig.1 (e) with **Participant** on the top and **Provider** on the bottom.

Example: The customer, the id is 3104, can be defined as follows.

$$Customer := (\lambda id_a : nat.Participant)3104 \tag{10}$$

1.4 Event

There are two basic events in SPDL, start event and end event. In Fig. 1 (f), the above one is the start event and below the end event. To support more other event in BPMN, we provide a constructor with an id as a parameter and the formalization of the event is present in Eq. 11.

$$Event := Start|End|(\lambda id_e : nat.\mu_e[\![EventIn]\!]) \tag{11}$$

1.5 Gateway

We have defined a basic gateway, the exc (the exclusive gateway) and more gateways can be defined by a constructor. Exc is present in Fig. 1 (g).

$$Gateway := Exc|(\lambda id_g.\mu_g[\![GatewayIn]\!]) \tag{12}$$

1.6 Line

There are 6 kinds of line (as Fig. 1 (b) from top to bottom):

- Creating line: this line usually connects from an activity to a kind resource (or data). It means the connected resource (or data) is generated by the connected activity.
- Consuming line: this line usually connects from an activity to a kind resource (or data). It means the connected resource (or data) is used up by the connected activity.
- Using line: this line usually connects from an activity to a kind resource (or data). It means the connected resource (or data) is used by the connected activity.
- Exclusively using line: this line usually connects from an activity to a kind resource (or data). It means the connected resource (or data) is used by the connected activity and at the same time this resource (data) cannot be used by other participants.
- Sequence line: this line usually connects from an activity to another one. It means the latter activity should be executed after the former one is finished and these two activities belong to the same participant.

– Message line: this line usually connects from an activity to another one. It means the latter activity should be executed after the former one is finished and these two activities do not belong to the same participant.

The first four lines are used to present the data flow and value flow. The last two lines, present the work flow, are formalized as follows:

$$\begin{aligned} LineP :=& \lambda lp_a : Activity.\mu_{lpa}[\![LinePA]\!] \\ & |\lambda lp_e : Event.\mu_{lpe}[\![LinePE]\!] \\ & |\lambda lp_g : Gateway.\mu_{lpg}[\![LinePG]\!] \\ Line :=& \lambda l_s : LineP \quad \lambda l_t : LineP.\mu_l[\![LineIn]\!] \end{aligned} \tag{13}$$

In Eq.13, $LineP$ is the port and $Line$ is the combination of two ports (source port and target port).

1.7 Activity

The activity notation is present in Fig. 1 (c). And the formalization of activity is shown in Eq. 14:

$$\begin{aligned} Activity :=& \lambda a_p : Actor. \\ & \lambda a_r : list(Op_r) \lambda a_d : list(Op_d) \\ & \mu_a[\![ActivityConstructor]\!] \end{aligned} \tag{14}$$

The first parameter is the actor, the second is the resource operation list (including four kinds) and the third is the data operation list.

Example: Upload video expressed as Eq.15, is an activity.

$$Upload := \mu_a(Customer\,(\nu_{cr} Video :: nil)\, nil) \tag{15}$$

1.8 Activity Pattern

Activity Pattern is an abstraction of the activity.

$$\begin{aligned} ActivityPattern :=& \lambda ap_p : Actor. \\ & \lambda ap_r : list(Op_r). \lambda ap_d : list(Op_d). \\ & \mu_{ap}[\![ActivityPatternConstructor]\!] \end{aligned} \tag{16}$$

Definition (Activity Pattern rule): An activity a adopts an activity Pattern ap if and only if they have the same actor and the latter's list of resource operations and data operations are sub-list of the former's.

$$\begin{aligned} ap \preceq a :=& (\sigma_{ap}(ap_p)) = (\sigma_a(a_p)) \\ & \times (\sigma_{ap}(ap_r)) \subseteq_{list} (\sigma_a(a_r)) \times (\sigma_{ap}(ap_d)) \subseteq_{list} (\sigma_a(a_d)) \end{aligned} \tag{17}$$

1.9 Business Rule

The business rules are defined in the type of *Product-dependent type*. Some useful are present as follows:

$$\begin{aligned}
\psi_{CAA}[\![ConnectAAProp]\!] &:= (\sigma_l(l_s) = \sigma_{lpa}(a_1)) \times (\sigma_l(l_t) = \sigma_{lpa}(a_2)) \\
&: \prod(l : Line) \prod(a_1, a_2 : Activity) Prop(l, a_1, a_2) \\
\psi_{CAE}[\![ConnectAEProp]\!] &:= (\sigma_l(l_s) = \sigma_{lpa}(a)) \times (\sigma_l(l_t) = \sigma_{lpe}(e)) \\
&: \prod(l : Line) \prod(a : Activity) \prod(e : Event) Prop(l, a, e) \\
\psi_{CEA}[\![ConnectEAProp]\!] &:= (\sigma_l(l_s) = \sigma_{lpe}(e)) \times (\sigma_l(l_t) = \sigma_{lpa}(a)) \\
&: \prod(l : Line) \prod(e : Event) \prod(a : Activity) Prop(l, e, a) \\
\psi_{AR}[\![ActivityRProp]\!] &:= (\sigma_a(a_r) \ni_{list} r) : \prod(a : Activity) \prod(r : Op_r) Prop(a, r) \\
\psi_{AD}[\![ActivityDProp]\!] &:= (\sigma_a(a_d) \ni_{list} d) : \prod(a : Activity) \prod(d : Op_d) Prop(a, d) \\
\psi_{A}[\![ActorProp]\!] &:= (\sigma_a(a_p) = p) : \prod(a : Activity) \prod(p : Actor) Prop(a, p)
\end{aligned} \tag{18}$$

The $ConnectAAProp$ is the proposition checking whether l connects from $a1$ to $a2$. The $ConnectAEProp$ and $ConnectEAProp$ check the connection from activity to event. The $ActivityRProp$ is the proposition checking whether a involves r as a parameter. In the similar way, $ActivityDProp$ are defined. The $ActorProp$ checks whether p is the actor of a.

All these propositions defined above are the basic business rules. Notation ψ is used to represent a rule.

In spite of basic business rule, we define a special rule, named **pattern rule** which means the rule constraints on the pattern. The pattern rule can be constructed by basic business rules while it helps to design a new business pattern. Because of limit of the paper volume, we will not discuss the equation of business rule and pattern rule, and the detailed method of designing a new pattern by pattern rules.

The activity pattern proposition is a pattern rule.

$$\begin{aligned}
\psi^*_{AP}[\![ActivityPatternProp]\!] &:= ap \preceq a \\
&: \prod(a : Activity) \prod(ap : ActivityPattern) Prop(a, ap)
\end{aligned} \tag{19}$$

References

1. J. Yin, Z. Luo, Y. Li, B. Fan, and Z. Wu, "Towards a service pattern model supporting quantitative economic analysis," in *SERVICES 2014 Workshops*, SERVICES 2014 Workshops, (Ancherage, Alaska, USA), 2014.