Rapport interne 2012-13-04-DI-STA

Langage de propriétés

Bilal Kanso, Safouan Taha

Résumé

– Grammaire du langage de propriétés

– Sémantique du langage de propriétés

Table des matières

1 Grammaire du langage

Pour définir la grammaire de notre langage d'expression de propriétés, nous ajoutons les règles EBNF présentées dans Table 1 à la syntaxe concrète d'OCL standardisée par l'OMG dans le document [1]. Dans ces règles, les non-terminaux sont désignés en *italique*, les terminaux sont en **gras** et la construction (...)? désigne une partie optionnelle. Les non-terminaux soulignés proviennent de la grammaire concrète d'OCL. Par exemple, Le terminal OclExpression désigne un prédicat OCL.

TempOCL	: :=	temp (name) ? ' :' *TempSpec*
TempSpec	: :=	*Quantif* ? *Pattern Scope*
Quantif	: :=	let Variable (',' Variable)* in
Pattern	: :=	always OclExpression
	|	never *Event*
	|	eventually *Event*((at least | at most) ? integer times) ?
	|	*EventChain* preceding (directly | strictly) ? *EventChain*
	|	*EventChain* following (directly | strictly) ? *EventChain*
Scope	: :=	globally
	|	before *Event* ('[' | ']') ?
	|	after ('[' | ']') ?*Event*
	|	between('[' | ']') ? last ? *Event* and *Event* ('[' | ']') ?
	|	after('[' | ']') ? last ? *Event* unless *Event* ('[' | ']') ?
	|	when OclExpression
Event	: :=	*CallEvent* ('|' *Event*) ?
	|	*ChangeEvent* ('|' *Event*) ?
EventChain	: :=	*Event* (',' *Event*)*
	|	*Event* (';' *Event*)*
CallEvent	: :=	isCalled '(' Operation (',' pre : OclExpression) ? (',' post : OclExpression) ? ')'
	|	isCalled '(' anyOp (',' pre : OclExpression) ?
		(',' post : OclExpression) ?')'
		(except *Event*) ?
ChangeEvent	: :=	becomesTrue '(' OclExpression ')'

TABLE 1 – Grammaire du langage de propriétés

2 Sémantique du langage de propriétés

Nous définissons ici formellement la sémantique du langage proposé. Les propriétés écrites dans notre langage sont interprétées le long des scénarios d'appels de méthodes d'un modèle objet.

2.1 Événement et Scénario

2.1.1 Événement

Nous définissons d'abord notre alphabet. Il s'agit de l'ensemble des événements atomiques.

Definition 1 (Alphabet des événements atomiques) *Soit $\mathscr{M}$ un modèle objet. Soient $\mathbb{O}$ l'ensemble de toutes les opération et $\mathbb{E}$ l'ensemble de toutes les expressions OCL. L'**alphabet des événements atomiques** Σ est défini par l'ensemble $\mathbb{O} \times \mathbb{E} \times \mathbb{E}$.*

Un événement atomique $e \in \Sigma$ prend la forme : $e = (op, pre, pos)$. Il représente l'invocation de l'opération op dans un contexte où la précondition pre est satisfaite dans l'état de départ, la postcondition pos est satisfaite dans l'état d'arrivée.

Nous définissons formellement la notion d'événement introduite dans notre extension temporelle du langage OCL.

Definition 2 (Événement) *Soit Σ l'alphabet des événements atomiques. Un **événement** est défini soit comme un* $\mathsf{isCalled}(op, pre, pos)$*, un* $\mathsf{isCalled}(op, pre, pos)$ *ou un* $\mathsf{becomesTrue}(\mathsf{e})$ *où :*

- $\mathsf{isCalled}((op, pre, pos))$ *est défini par l'ensemble :*

$$\{(op, pre', pos') \in \Sigma \mid pre' \Longrightarrow pre, pos' \Longrightarrow pos\}$$

- $\mathsf{isCalled}((op, pre, pos))$ *est défini par l'ensemble :*

$$\{(op, pre', pos') \in \Sigma \mid pre' \Longrightarrow pre, pos' \Longrightarrow pos\}$$

- $\mathsf{becomesTrue}(e)$ *est défini par l'ensemble :*

$$\{(op, pre, pos) \in \Sigma \mid op \in \mathbb{O}, pre \Longrightarrow \neg e, pos \Longrightarrow e\}$$

Définition 2 appelle à trois remarques :

- L'opération op peut être remplacée par le mot clé anyOp qui désigne l'ensemble de toutes les opérations

$$\mathsf{isCalled}(\mathsf{anyOp}, pre, post) = \bigcup_{op \in \mathbb{O}} \{(op, p, q) \in \Sigma \mid p \Rightarrow pre,\ q \Rightarrow post\}$$

$$\mathsf{isCalled}(\mathsf{anyOp}, pre, post) = \bigcup_{op \in \mathbb{O}} \{(op, p, q) \in \Sigma \mid p \Rightarrow pre,\ q \Rightarrow post\}$$

- L'événement $\mathsf{becomesTrue}(e)$ est équivalent à l'événement $\mathsf{isCalled}(\mathsf{anyOp}, \neg e, e)$. Nous choisissons de l'introduire comme primitive dans notre langage pour en faciliter l'utilisation.

- Un événement ne représente pas un seul événement atomique, mais un sous-ensemble d'événements atomiques. Cela est principalement dû à l'introduction d'une précondition et d'une postcondition dans la définition d'un événement. Intuitivement, $\mathsf{isCalled}((op, pre, pos))$ est l'ensemble de tous les événements atomiques dans lesquels l'opération op est impliquée, non seulement dans le contexte de la précondition pre et de la postcondition pos, mais aussi dans le contexte de toute précondition et toute postcondition satisfaisant pre et pos respectivement. Le même constat s'applique sur l'utilisation des étiquettes.
  On notera 2^{Σ} l'ensemble de tous les événements.

Definition 3

- *L'**opérateur de disjonction** des événements $|$ est défini comme suivant :*

$$\begin{array}{rccc} | : & 2^{\Sigma} \times 2^{\Sigma} & \longrightarrow & 2^{\Sigma} \\ & (e_1, e_2) & \longmapsto & e_1 \cup e_2 \end{array}$$

- *L'**opérateur d'exclusion** except est défini comme suivant :*

$$\begin{array}{rccc} \mathsf{except} : & 2^{\Sigma} \times 2^{\Sigma} & \longrightarrow & 2^{\Sigma} \\ & (e_1, e_2) & \longmapsto & e_1 \setminus e_2 \end{array}$$

Ces définitions sont amplement facilitées par notre représentation ensembliste des événements.

2.1.2 Scénario

Nous introduisons la notion d'un scénario permettant d'interpréter les expressions temporelles d'OCL. Un *scénario* σ dans un modèle $\mathscr{M}$ est une séquence finie $(e_0,\ldots,e_n) \in \Sigma^*$ d'événements atomiques. Un tel scénario reflète un ordre temporel entre les déclenchements d'événements, où la notion du temps est implicitement spécifiée. Dans un scénario $(e_0,e_1,\ldots,e_n)$, il y a un instant logique associé à l'événement e_0 qui précède l'instant logique associé à l_1, et ainsi de suite. Cette introduction implicite du temps dans le scénario nous permet d'exprimer des propriétés temporelles, faisant référence au temps linéaire discret.

Dans la suite, pour tout scénario $\sigma \in \Sigma^*$ de longueur n, nous écrivons $\sigma = (\sigma(0),\ldots,\sigma(n-1))$. Ainsi, la notation $\sigma(i)$ représente l'élément qui se trouve à l'indexe i et $\sigma(i:j)$ représente le segment de σ contenant les éléments entre i et j.

2.2 Expression temporelle

Par la suite, nous donnons la sémantique des propriétés écrites avec notre extension temporelle d'OCL. Les propriétés sont interprétées le long d'un scénario d'exécution du modèle objet $\mathscr{M}$.

Definition 4 (Sémantique des scopes) *Soit $\mathbb{S}$ l'ensemble des scopes définis dans la grammaire présentée dans Table 1. La **sémantique d'un scope** $s \in \mathbb{S}$ est donnée par la fonction* [1]

$$[[s]]^s : \Sigma^* \longrightarrow 2^{\Sigma^*}$$

définie pour tout scénario $\sigma \in \Sigma^*$ de longueur n comme suivant :

- $[[\mathsf{globally}]]^s(\sigma) = \{\sigma\}$
- $[[\mathsf{before}\ E\ a]]^s(\sigma) = \{\sigma(0:i-j) \mid \sigma(i) \in E\ et\ \forall k, 0 \leq k < i, \sigma(k) \notin E\}$

  où

  $$j = \begin{cases} 1 & si \quad a = [\\ 0 & si \quad a =] \end{cases}$$

- $[[\mathsf{After}\ a\ E]]^s(\sigma) = \{\sigma(i+j:n-1) \mid \sigma(i) \in E\ et\ \forall k, 0 \leq k < i, \sigma(k) \notin E\}$

  où

  $$j = \begin{cases} 0 & si \quad a = [\\ 1 & si \quad a =] \end{cases}$$

- $[[\mathsf{between}\ a\ c\ E_1\ \mathsf{and}\ E_2\ b]]^s(\sigma) = \{\sigma(i_k+\alpha : j_k-\beta) \mid$
  $\forall k \geq 0, i_k < j_k < i_{k+1}, \sigma(i_k) \in E_1, \sigma(j_k) \in E_2,$
  $\forall m, i_k < m < j_k, \sigma(m) \notin \theta\ et$
  $\forall l, j_k < l < i_{k+1}, \sigma(l) \notin E_1\}$

  où

  $$\alpha = \begin{cases} 0 & si \quad a = [\\ 1 & si \quad a =] \end{cases} \qquad \beta = \begin{cases} 0 & si \quad b =] \\ 1 & si \quad b = [\end{cases} \qquad \theta = \begin{cases} E_2 & si \quad c = \text{""} \\ E_1 \cup E_2 & si \quad c = \mathsf{last} \end{cases}$$

- $[[\mathsf{after}\ a\ c\ E_1\ \mathsf{unless}\ E_2\ b]]^s(\sigma) =$
  $\{\sigma(i_k+1 : j_k-1) \mid \forall k \geq 0, i_k < j_k < i_{k+1}, \sigma(i_k) \in E_1, \sigma(j_k) \in E_2,$
  $\forall m, i_k < m < j_k, \sigma(m) \notin \theta\ et$
  $\forall l, j_k < l < i_{k+1}, \sigma(l) \notin E_1\}$
  $\cup \{\sigma(i : n-1) \mid \sigma(i) \in E_1, \forall m \geq i, \sigma(m) \notin E_2\}$

1. 2^X désigne l'ensemble des parties de l'ensemble X.

où

$$\alpha = \begin{cases} 0 & si \quad a = [\\ 1 & si \quad a =] \end{cases} \qquad \beta = \begin{cases} 0 & si \quad b =] \\ 1 & si \quad b = [\end{cases} \qquad \theta = \begin{cases} E_2 & si \quad c = \text{""} \\ E_1 \cup E_2 & si \quad c = \text{last} \end{cases}$$

– $[[\text{when } P]]^s(\sigma) = [[\text{after }] \text{ becomesTrue}(P) \text{ unless becomesTrue}(\neg P) [\ ]]^s(\sigma)$

Definition 5 (Fonctions de matchings) *Soit $E_1 \dots E_m$ une séquence d'événements (pas nécessairement consécutifs). On distingue une séquence d'ordre strict notée $[E_1;\dots;E_m]$ d'une séquence d'ordre non strict $[E_1,\dots,E_m]$. Nous définissons deux **fonctions de matching** qui repèrent le début ou la fin d'une séquence*

– $matchStart(\sigma, E_1 \dots E_m, i) \Longleftrightarrow \begin{cases} \exists 0 < i_2 < i_3 < \dots < i_m \; si \; E_1 \dots E_m = [E_1;\dots;E_m] \\ \exists 0 \leq i_2 \leq i_3 \leq \dots \leq i_m \; si \; E_1 \dots E_m = [E_1,\dots,E_m] \end{cases}$
$tel \; que \; \sigma(i) \in E_1, \; \sigma(i+i_2) \in E_2, \dots, \; \sigma(i+i_m) \in E_m$

– $matchEnd(\sigma, E_1 \dots E_m, i) \Longleftrightarrow \begin{cases} \exists 0 < i_2 < i_3 < \dots < i_m \; si \; E_1 \dots E_m = [E_1;\dots;E_m] \\ \exists 0 \leq i_2 \leq i_3 \leq \dots \leq i_m \; si \; E_1 \dots E_m = [E_1,\dots,E_m] \end{cases}$
$tel \; que \; \sigma(i) \in E_m, \; \sigma(i-i_2) \in E_{m-1}, \dots, \; \sigma(i-i_m) \in E_1$

Definition 6 (Sémantique des patterns) *Soit $\mathbb{P}$ l'ensemble de tous les patterns définis dans la grammaire présentée dans Table 1. La **sémantique d'un pattern** $p \in \mathbb{P}$ est donnée par la fonction*

$$[[p]]^p : \Sigma^* \longrightarrow \{true, false\}$$

définie pour tout $\sigma \in \Sigma^*$ comme suivant :

– $[[\text{never } E]]^p(\sigma) \Leftrightarrow \forall i \geq 0, \sigma(i) \notin E$

– $[[\text{always } P]]^p(\sigma) \Leftrightarrow [[\text{never}(\text{isCalled}(\text{anyOp}, \_, \neg P))]]^p(\sigma)$

– $[[E_1 \text{ preceding } a \; E_2]]^p(\sigma) \Leftrightarrow \forall i \geq 0, (\sigma(i) \in E_2 \Rightarrow \alpha)$

où

$$\alpha = \begin{cases} \exists k \leq i, \sigma(k) \in E_1 & si \quad a = \text{""} \\ \exists k < i, \sigma(k) \in E_1 & si \quad a = \text{stricly} \\ \sigma(i-1) \in E_1 & si \quad a = \text{directly} \end{cases}$$

– $[[E_1 \text{ following } a \; E_2]]^p(\sigma) \Leftrightarrow \forall i \geq 0, (\sigma(i) \in E_2 \Rightarrow \alpha)$

où

$$\alpha = \begin{cases} \exists k \geq i, \sigma(k) \in E_1 & si \quad a = \text{""} \\ \exists k > i, \sigma(k) \in E_1 & si \quad a = \text{stricly} \\ \sigma(i+1) \in E_1 & si \quad a = \text{directly} \end{cases}$$

– $[[\text{eventually } E \; \alpha \text{ times}]]^p(\sigma) \Leftrightarrow \text{card}\big(\{i \mid \sigma(i) \in E\}\big) \begin{cases} \geq 1 & if \quad \alpha = \text{""} \\ = k & si \quad \alpha = k \\ \geq k & si \quad \alpha = \text{at least } k \\ \leq k & si \quad \alpha = \text{at most } k \end{cases}$

– $[[E_1 \dots E_l \text{ preceding } a\ F_1 \dots F_m]]^p(\sigma) \Leftrightarrow \forall i \geq 0, (matchStart(\sigma, F_1 \dots F_m, i) \Rightarrow \alpha)$

où

$$\alpha = \begin{cases} \exists k \leq i, matchEnd(\sigma, E_1 \dots E_l, k) & si \quad a = \text{""} \\ \exists k < i, matchEnd(\sigma, E_1 \dots E_l, k) & si \quad a = \text{stricly} \\ matchEnd(\sigma, E_1 \dots E_l, i-1) & si \quad a = \text{directly} \end{cases}$$

– $[[E_1 \dots E_l \text{ following } a\ F_1 \dots F_m]]^p(\sigma) \Leftrightarrow \forall i \geq 0, (matchEnd(\sigma, F_1 \dots F_m, i) \Rightarrow \alpha)$

où

$$\alpha = \begin{cases} \exists k \geq i, matchStart(\sigma, E_1 \dots E_l, k) & si \quad a = \text{""} \\ \exists k > i, matchStart(\sigma, E_1 \dots E_l, k) & si \quad a = \text{stricly} \\ matchStart(\sigma, E_1 \dots E_l, i+1) & si \quad a = \text{directly} \end{cases}$$

Definition 7 (Sémantique d'une expression temporelle) *Soient* $\mathbb{S}, \mathbb{P}$ *les ensembles des scopes et des patterns respectivement. Soit* $\sigma \in \Sigma^*$. *La* ***sémantique d'une expression temporelle d'OCL*** $(\text{pattern}, \text{scope}) \in \mathbb{P} \times \mathbb{S}$ *définie sur* σ, *que l'on dénote par* $\sigma \vDash (\text{pattern}, \text{scope})$, *est définie comme suivant :*

$$\sigma \vDash (\text{pattern}, \text{scope}) \Longleftrightarrow \forall \sigma' \in [[\text{scope}]]^s(\sigma),\ [[\text{pattern}]]^p(\sigma')$$

Bibliographie

[1] Object Management Group, Inc., editor. *OCL 2.2 (Final Adopted specification)*. http ://www.omg.org/spec/OCL/2.2/PDF, february 2010.